

Philippe Kruchten
Peggy Gregory (Eds.)

LNBIP 489

Agile Processes in Software Engineering and Extreme Programming – Workshops

XP 2022 Workshops, Copenhagen, Denmark, June 13–17, 2022
and XP 2023 Workshops, Amsterdam, The Netherlands, June 13–16, 2023
Revised Selected Papers


 Springer


OPEN ACCESS


Lecture Notes in Business Information Processing

489


Series Editors

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

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

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

Clemens Szyperski, *Microsoft Research, Redmond, WA, 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.

Philippe Kruchten · Peggy Gregory
Editors

Agile Processes in Software Engineering and Extreme Programming – Workshops

XP 2022 Workshops, Copenhagen, Denmark, June 13–17, 2022
and XP 2023 Workshops, Amsterdam, The Netherlands, June 13–16, 2023
Revised Selected Papers

Editors

Philippe Kruchten 
University of British Columbia
Vancouver, BC, Canada

Peggy Gregory 
University of Glasgow
Glasgow, UK



ISSN 1865-1348

ISSN 1865-1356 (electronic)

Lecture Notes in Business Information Processing

ISBN 978-3-031-48549-7

ISBN 978-3-031-48550-3 (eBook)

<https://doi.org/10.1007/978-3-031-48550-3>

© The Editor(s) (if applicable) and The Author(s) 2024. 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.

The use of general descriptive names, registered names, trademarks, service marks, etc. in this publication does not imply, even in the absence of a specific statement, that such names are exempt from the relevant protective laws and regulations and therefore free for general use.

The publisher, the authors, and the editors are safe to assume that the advice and information in this book are believed to be true and accurate at the date of publication. Neither the publisher nor the authors or the editors give a warranty, expressed or implied, with respect to the material contained herein or for any errors or omissions that may have been made. The publisher remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

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

Paper in this product is recyclable.

Preface

This volume contains papers from the research workshops presented at XP 2022 and XP 2023, respectively the 23rd and 24th International Conferences on Agile Software Development, held on June 13–17, 2022 at the IT University of Copenhagen, Denmark and June 13–16, 2023 in Amsterdam, the Netherlands.

XP is the premier agile software development conference combining research and practice. It is a unique forum where agile researchers, practitioners, thought leaders, coaches, and trainers get together to present and discuss their most recent innovations, research results, experiences, concerns, challenges, and trends. XP conferences provide an informal environment to learn and trigger discussions and welcome both people new to agile and seasoned agile practitioners.

The XP 2022 and XP 2023 research papers were published in the conference proceedings, LNBIP volumes 445 and 475. This companion volume, published after the conferences, contains selected revised workshop papers.

The research workshops provide a highly relevant, friendly, and interactive platform to share and discuss emerging and late breaking research findings as well as educational experiments and experiences. They represent smaller, close communities of passionate, emerging, and established researchers and a psychologically safe environment to provide and receive feedback. The publication of the post-conference proceedings allows the researchers and educators to fold into their papers the feedback and lessons learned from their participation in the conference and workshop sessions.

In 2022, the following three workshops took place:

- 3rd International Workshop on Agility with Microservices Programming
- 2nd International Workshop on Agile Sustainability
- Agile and Education

In 2023, six workshops were held:

- Workshop on Organisational Debt and Large-Scale Agile
- Workshop on Software-Intensive Business
- Workshop on Global and Hybrid Work
- Workshop on Fear-Based Agile Transformation
- Workshop on AI-assisted Agile
- Workshop on Agile-Quantum Software Engineering

In 2022, 6 workshop papers were accepted for publication in these post-proceedings, out of 11 submissions, and in 2023, 15 papers were accepted for publication out of 38 submissions. The review cycles used single-blind reviews using EasyChair.

In addition to the workshop papers these post-conference proceedings include a summary of a panel discussion on the future of hybrid work.

We would like to extend our sincere thanks to all the people who contributed to XP2022 and XP2023: the authors, reviewers, chairs, and volunteers. Finally, we would

like to express our gratitude to the XP Conference Steering Committee and the Agile Alliance for their ongoing support.

June 2023

Peggy Gregory
Philippe Kruchten

Organization

Conference Co-chairs

Wouter Lagerweij	Lagerweij Consultancy, The Netherlands
Suzanne Lagerweij	Lagerweij Consultancy, The Netherlands

Program Co-chairs

Christoph J. Stettina	Leiden University/Centre for Innovation, The Netherlands
Juan Garbajosa	Universidad Politécnica de Madrid, Spain

Workshop Chair

Peggy Gregory	University of Glasgow, UK
---------------	---------------------------

Publication Co-chairs

Philippe Kruchten	University of British Columbia, Canada
Peggy Gregory	University of Glasgow, UK

Third International Workshop on Agility and Microservices

Florian Rademacher	University of Dortmund, Germany
Eduardo Guerra	Free University of Bozen-Bolzano, Italy
Larisa Safina	Inria Lille - Nord Europe, France

2nd International Workshop on Agile Sustainability

Coral Calero Munoz	Universidad de Castilla-La Mancha, Spain
Juan Garbajosa	Universidad Politécnica de Madrid, Spain
Jennifer Perez	Universidad Politécnica de Madrid, Spain
Agustin Yagüe	Universidad Politécnica de Madrid, Spain

Education and Training Track

Maarit Laanti

Nitor Delta, Finland

Martin Kropp

University of Northwestern Switzerland

Workshop on Organisational Debt and Large-Scale Agile Software Development

Muhammad O. Ahmad

Karlstad University, Sweden

Tomas Gustavsson

Karlstad University, Sweden

Workshop on Software-Intensive Business

Jorge Melegati

Free University of Bozen-Bolzano, Italy

Karl Werder

University of Cologne, Germany

Dimitrik Petrik

University of Stuttgart, Germany

Workshop on Global and Hybrid Work in SW Engineering

Maria Paasivaara

LUT University & Aalto University, Finland

Xiaofeng Wang

Free University of Bozen-Bolzano, Italy

Workshop on Fear-Based Agile Transformations

Julian Bass

University of Salford, UK

Morten Elvang

Accenture, Denmark

Workshop on AI-assisted Agile

Pekka Abrahamsson

Tampere University, Finland

Anh Nguyen-Duc

University of South-Eastern Norway, Norway

Workshop on Agile-Quantum Software Engineering

Muhammad Azeem Akbar
Arif Ali Khan

LUT University, Finland
University of Oulu, Finland

Program Committee

Pekka Abrahamsson
Steve Adolph
Ademar Aguiar
Scott Ambler
Craig Anslow
Leonor Barocca
Hubert Baumeister
Jan Bosch
Frank Buschmann
Daniela N. Cruzes
Torgeir Dingsøy

Tampere University, Finland
cprime, Canada
University of Porto, Portugal
SA+A, Canada
Victoria University of Wellington, New Zealand
Open University, UK
Technical University of Denmark
Chalmers University of Technology, Sweden
Siemens AG, Germany
NTNU, Norway
Norwegian University of Science and Technology,
Norway

Yael Dubinsky
Jutta Eckstein
Henry Edison
Ilenia Fronza
Juan Garbajosa
Alfredo Goldman
Peggy Gregory
Eduardo Guerra
Tomas Gustavsson
Helena Holmström Olsson
Kiyoshi Honda
Fabio Kon
Philippe Kruchten
Thomas Kude
Marco Kuhrmann
Casper Lassenius
Ville Leppänen
Lech Madeyski

Kinneret Academic College, Israel
Independent, Germany
Blekinge Institute of Technology, Sweden
Free University of Bozen-Bolzano, Italy
Universidad Politécnica de Madrid, Spain
University of São Paulo, Brazil
University of Glasgow, UK
Free University of Bozen-Bolzano, Italy
Karlstads Universitet, Sweden
University of Malmö, Sweden
Osaka Institute of Technology, Japan
University of São Paulo, Brazil
University of British Columbia, Canada
University of Bamberg, Germany
Reutlingen University, Germany
Aalto University, Finland
University of Turku, Finland
Wroclaw University of Science and Technology,
Poland
PUCRS, Brazil
University of Oslo, Norway
University of Calgary, Canada

Sabrina Marczak
Antonio Martini
Frank Maurer

Tommi Mikkonen	University of Helsinki, Finland
Alok Mishra	Atilim University, Turkey
Nils Brede Moe	SINTEF, Norway
Parastoo Mohagheghi	Norwegian Labour and Welfare Administration, Norway
Jürgen Münch	Reutlingen University, Germany
Anh Nguyen-Duc	University of South-Eastern Norway, Norway
Tyron Offerman	Leiden University, The Netherlands
Maria Paasivaara	LUT University & Aalto University, Finland
Cécile Péraire	Carnegie Mellon University, USA
Rafael Prikladnicki	PUCRS, Brazil
Adam Przybyłek	Gdansk University of Technology, Poland
Pilar Rodríguez	Universidad Politécnica de Madrid, Spain
Helen Sharp	Open University, UK
Darja Šmite	Blekinge Institute of Technology, Sweden
Simone Spiegler	Monash University, Australia
Christoph J. Stettina	Leiden University/Centre for Innovation, The Netherlands
Viktoria Stray	University of Oslo, Norway
John F. Tripp	Clemson University, USA
Rini Vansolingen	Delft University of Technology, The Netherlands
Joost Visser	Leiden University, The Netherlands
Stefan Wagner	University of Stuttgart, Germany
Xiaofeng Wang	Free University of Bozen-Bolzano, Italy
Hironori Washizaki	Waseda University, Japan
Agustin Yagüe	Universidad Politécnica de Madrid, Spain

Additional Reviewers

Halimeh Agh	University of Stuttgart, Germany
Pavel Nedvědický	University of Stuttgart, Germany
Ana Moises de Souza	Norwegian University of Science and Technology, Norway
Eva Zimmermann	University of Stuttgart, Germany

Steering Committee

Hubert Baumeister	Technical University of Denmark, Denmark
François Coallier	Ecole de technologie supérieure, Canada
Jutta Eckstein	Independent, Germany

Teresa Foster	Agile Alliance, USA
Juan Garbajosa (Chair)	Universidad Politécnica de Madrid, Spain
Peggy Gregory	University of Glasgow, UK
Wouter Lagerweij	Lagerweij Consultancy, The Netherlands
Casper Lassenius	Aalto University, Finland
Maria Paasivaara	LUT University & Aalto University, Norway
Viktoria Stray	University of Oslo, Norway

Sponsoring Organization

Teresa Foster	Agile Alliance, USA
---------------	---------------------

Contents

2nd International Workshop on Agile Sustainability

Connecting Agile with Theory of Change	3
<i>Jutta Eckstein and Steve Holyer</i>	
Enhancing Agile Software Development Sustainability Through the Integration of User Experience and Gamification	12
<i>Manal Alhammad and Ana Moreno</i>	
Sustainable IT in an Agile DevOps Setup Leads to a Shift Left in Sustainability Engineering	21
<i>Alexander Poth, Daniela Eißfeldt, Christian Heimann, and Stefan Waschk</i>	

3rd International Workshop on Agility and Microservices

Improving the Implementation of Microservice-Based Systems with Static Code Analysis	31
<i>Sebastian Copei, Maximilian Schreiter, and Albert Zündorf</i>	
Towards an Architecture-Centric Methodology for Migrating to Microservices	39
<i>Jonas Fritzsich, Justus Bogner, Markus Haug, Stefan Wagner, and Alfred Zimmermann</i>	

Agile in Education Tack

Being Agile in a Data Science Project	51
<i>Renato Cordeiro, Isaque Alves, Samara Alves, and Alfredo Goldman</i>	

Panel

The Future of Work: Agile in a Hybrid World	63
<i>Dennis Mancl and Steven D. Fraser</i>	

Organisational Debt and Large-Scale Agile

Organizational Debt in Large-Scale Hybrid Agile Software Development:
A Case Study on Coordination Mechanisms 75
Zixuan Liu, Viktoria Stray, and Tor Sporsem

Software-Intensive Business

The Know-How of Agile Retrospectives in Software Startups 87
Dron Khanna and Xiaofeng Wang

Analytics Practices in Practice: How Software Startup Companies Are
Applying Analytics? 97
Usman Rafiq, Xiaofeng Wang, and Luciana Zaina

Towards a X-as-a-Service Application in Industrial Laundry – A Case
Study of Information Requirement Engineering in Emerging Data
Ecosystems 107
Maximilian Werling, Alexandra Keller, and Heiner Lasi

Software Startup Ecosystem in Namibia 116
Hilma Aludhilu and Erkki Sutinen

Industry Expectations for Product Ops Professionals: A Review of Job
Advertisements 125
Bogdan Moroz, Andrey Saltan, and Sami Hyrynsalmi

Global and Hybrid Work in Software Engineering

Unveiling the Spectrum of Hybrid Work in Software Engineering:
Research Directions 139
Maria Paasivaara and Xiaofeng Wang

Defining a Remote Work Policy: Aligning Actions and Intentions 149
Darja Smite and Nils Brede Moe

Fear-Based Agile Transformations

Business Development in Large-Scale Agile Software Development:
Barriers and Enablers 161
John Olav Olsen, Viktoria Stray, and Nils Brede Moe

AI-assisted Agile

ChatGPT as a Tool for User Story Quality Evaluation: Trustworthy Out of the Box? 173
Krishna Ronanki, Beatriz Cabrero-Daniel, and Christian Berger

Survey of AI Tool Usage in Programming Course: Early Observations 182
Mika Saari, Petri Rantanen, Mikko Nurminen, Terhi Kilamo, Kari Systä, and Pekka Abrahamsson

Turning Large Language Models into AI Assistants for Startups Using Prompt Patterns 192
Xiaofeng Wang, Mohammad Idris Attal, Usman Rafiq, and Sylvia Hubner-Benz

ChatGPT as a Fullstack Web Developer - Early Results 201
Pekka Abrahamsson, Tatu Anttila, Jyri Hakala, Juulia Ketola, Anna Knappe, Daniel Lahtinen, Väinö Liukko, Timo Poranen, Topi-Matti Ritala, and Manu Setälä

Agile-Quantum SE 2023

Reviewing Crypto-Agility and Quantum Resistance in the Light of Agile Practices 213
Lodovica Marchesi, Michele Marchesi, and Roberto Tonelli




Empirical Investigation of Quantum Computing on Solving Complex Problems 222
Shahid Hussain, Yuba Neupane, Wen-Li Wang, Naseem Ibrahim, Saif Ur Rehman Khan, and Asif Kareem

Author Index 231

2nd International Workshop on Agile Sustainability



Connecting Agile with Theory of Change

Jutta Eckstein¹  and Steve Holyer²  

¹ Gaußstr. 29, 38106 Braunschweig, Germany
jutta@jeckstein.com

² Engage Results, Aemlerstrasse 114, 8003 Zurich, Switzerland
coach@engageresults.com

Abstract. The majority of non-profit organizations, including those in the sustainability sector, use Theory of Change to define, plan, and evaluate their change initiatives [14, 26]. “Theory of Change is essentially a comprehensive description and illustration of how and why a desired change is expected to happen in a particular context.” [5] Both, Theory of Change and Agile acknowledge that before a plan is validated by implementation, everything is an assumption and all assumptions may have to change until validated by delivery. Therefore, planning and evaluation must both be responsive to change, and both must incorporate new learning. This paper presents an overview of Theory of Change and examines how it can align and connect with Agile. The agile community is starting to take more responsibility for sustainability, and it is beginning to support the sustainability sector as the second appearance of the “Workshop on Agile Sustainability” at the XP 2022 conference shows. However, when supporting the sustainability sector, agile practitioners are likely to encounter Theory of Change, and they will need to understand it. Thus, this paper provides an overview of Theory of Change and identifies connections between the Theory of Change approach and agile mindsets, methods, and practices. This will especially help and encourage agile practitioners to support the sustainability sector.

Keywords: Agile Manifesto · Change · Complexity · Principles · Theory-of-Change · Sustainability · Values

1 Introduction

Agile values, principles, and practices can be used to support the sustainability sector. However, supporting this sector also requires learning about its values, principles, and practices. An approach that offers these connections between Agile and the sustainability sector values, principles, and practices is the so-called “Theory of Change”. This approach is frequently used in non-profit organizations, non-governmental organizations (NGOs), and other non-commercial entities which make up a significant part of the sustainability sector. A quick review of the literature on Theory of Change makes it clear that an effective Theory of Change must be responsive within a complex adaptive

J. Eckstein—Independent.

© The Author(s) 2024

P. Kruchten and P. Gregory (Eds.): XP 2022/2023 Workshops, LNBI 489, pp. 3–11, 2024.

https://doi.org/10.1007/978-3-031-48550-3_1

system because the sustainability sector is dealing with complex issues [5, 27]. The same review indicates that the implementation of Theory of Change often takes a more linear approach, indulging in what the eXtreme Programming community might call “big design up front”.

In this paper, we introduce Theory of Change to agile practitioners and show how Agile and Theory of Change (or rather the sustainability sector) can relate and mutually benefit each other. In the next sections, we provide an overview of Theory of Change, before we connect Theory of Change with Agile. We will provide some examples, followed by a discussion and conclusion section, and we will finalize the paper with the references.

2 Theory of Change

The Aspen Institute Roundtable on Community Change developed Theory of Change as an approach for evaluating community-based change programs for “a free, just, and equitable society” [29].

The benevolent organization Comic Relief in the UK commissioned an early study on Theory of Change and how it is used by non-profit organizations. That study shows how Theory of Change can be used to evaluate change as well as to plan for it and guide it. Based on this understanding, the Center for Theory of Change gives this definition: “Theory of Change is essentially a comprehensive description and illustration of how and why a desired change is expected to happen in a particular context.” [5].

“Outlining a Theory of Change involves at its most basic making explicit a set of assumptions in relation to a given change process.” [27]. Thus, a common question asked in the non-profit sector is “What is your theory of change?” The question, in fact, asks for the underlying assumptions of a change program. These assumptions are both a way to communicate program goals to funders and can also serve to promote internal learning on program strategy, according to Valters [27]. Since the aid and development sector is operating in a complex space, it needs to be understood that a Theory of Change will need to change over time as more is learned during a change initiative.

The core of a Theory of Change is often “if-then” statements with high- and low-order goals, that are followed by a roadmap that typically includes vision, mission, and the program structure [10]. The first part of the statement, introduced by “if” declares the main assumption versus the second part, introduced by “then” which describes the assumed outcome. This is accompanied by metrics for evaluating each outcome.

From an agile perspective, this resonates with Behavioral Driven Development, which is following the “given - when - then” Syntax. In that case, “given” introduces the actual situation (which might be an assumption), “when” declares the actions that will lead to the assumed outcomes that are defined by “then” [24]. In this sense, you can associate Theory of Change with a (behavioral) test-driven approach.

Theory of Change typically uses a forward-thinking approach to change. As Rogers points out, “[t]he future perspective is used to create an optimistic framework to tackle difficult and complex problems [...]” [23]. Instead of looking at past data to create immediate changes (as it is practiced in Agile in retrospectives, [7]), Theory of Change builds a model from the future. It takes what planners imagine will have happened in the

future and builds backward on that. While Agile retrospectives aim to improve the future by learning from history, there is also an Agile approach known as the Futurespective, where you imagine the future and look for insights and actions that will secure that future vision [15].

3 Connecting Theory of Change and Agile

Valters lists four core principles for developing or refining Theory of Change. These simple rules provide focus and deal with the complexity of change: Focus on the process, Prioritize learning, Be locally-led, and Think compass, not map [27].

3.1 Focus on the Process

This principle acknowledges that as important as it is to set off with a Theory of Change, it is more important to continuously uncover assumptions because many assumptions will not be detected at the start of the change [27]. This is clearly meant to validate the process and therefore aligns with the first and last value statement of the Agile Manifesto, “Individuals and interactions over processes and tools.” and “Responding to change over following a plan” [2].

To find the mutual benefits, we first look at the connections we can make. Focus on the process is related to agile planning and even more so to Agile Chartering [17]. In both cases, it is acknowledged that the act of the process (the planning and chartering) is more important than the outcome (the plan or the charter). To be effective in a complex system an Agile Charter and Theory of Change must be continuously revisited, and are always “good enough for now” [17, 28]. There are many ways to create an Agile Charter. Liftoff is a framework for chartering that identifies 3 focus areas of an Agile Charter (Purpose, Alignment, and Context). Each focus area also has three components [17]:

- Chartering for purpose means creating a definition of the work to be done that is both inspiring and testable.
- Chartering for context means creating a shared awareness of the larger complex adaptive system in which a team or project group operates. One place Agile Chartering with Liftoff shines is when it helps the team visualize the dynamically changing relationships and interplay of stakeholders and the team.
- Chartering for Alignment focuses on a group of people (a team or a project group) establishing or deepening their alliance so they can work towards the purpose within the greater context.

3.2 Prioritize Learning

It is important to understand what can be learned from every planned activity. Theory of Change is grounded on assumptions, which means learning has to be at the core to understand if these assumptions have to change based on what is really happening. The following questions prioritize learning [27]:

- Learning for what? This first question focuses on the purpose of the learning and thus makes this purpose of the change transparent.

- Learning for whom? This second question explains who will mainly benefit from the learning.
- What kind of learning? The last question ensures that there will be a critical reflection on the assumptions. Thus, this question asks explicitly for double-loop and not single-loop learning, to understand the reasoning for what happened [3].

Similarly, Agile features continuous learning. The last principle of the Agile Manifesto is stated, “At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behavior accordingly.” It is understood as asking agile teams to learn continuously. This learning is practiced in Agile retrospectives. A team “adjusting its behavior” refers to single-loop learning, yet, as Eckstein stated: “[...] if the team uses the retrospectives additionally for examining the support and effect of (organizational) norms on its own behavior also double-loop learning is happening.” [9] Moreover, in a retrospective, the team decides on actions, assuming that these actions will change the situation for the better. Using a Theory-of-Change-lens would mean validating these assumptions in the next retrospective.

The three questions (learning for what?, whom?, and what kind?) also relate to the concept of user stories, where the focus is on what problem needs to be solved, who will benefit from it, and what is the purpose/reason for the story. Or as Jeff Patton expressed it: “Good story conversations are about who and why not just what” [21]. Looking at stories from a Theory of Change perspective shows that stories are in fact also assumptions of the fulfillment of customers’ needs. The assumptions can only be confirmed by observing the outcomes of delivering the story.

3.3 Be Locally-Led

Particularly in the aid and development sector, it is important not to impose change on people. People from developed countries should never pretend to know more than someone from a developing country about their own needs and necessary changes. Thus, supporting people in developing countries always means that the change is locally led by the people in those countries [27].

There are several examples of this principle of local leadership in Agile. In general, self-organization is a core agile concept that ensures the people who are involved are also the ones who act. The Agile Manifesto points this out in various ways [2]. For example, the first value statement, “Individuals and interactions over processes and tools” emphasizes that it is valuable for the people doing the work to decide how they work. Or the eleventh principle, “The best architectures, requirements, and designs emerge from self-organizing teams.” clarifies the value of self-organization in achieving great results. Or as Woody Zuill says, “This is an important concept for us: The team doing the work can best determine how to do that work.” [31].

Moreover, Open Space Technology (OST) is both a facilitation technique and an approach for organizational development, problem-solving, and change that relies on self-organization as a driver for dealing with complexity and complex issues in an open complex adaptive system [20]. OST is applied by agile practitioners for agile transitions, company-wide agility, and for addressing many other complex challenges [8, 18, 19]. OST approaches complexity by inviting everyone with a stake in a big question or

initiative to gather and offer questions and topics building a flexible agenda for a working meeting. Then, everyone gathered can choose how to organize to tackle all of the most important questions and topics [20]. This contrasts with an approach where leaders and/or managers decide what and how something needs to be done by the staff.

Finally, Extreme Programming suggests having a customer on-site as a team member, to avoid imposing features on the customer or working with invalidated assumptions [4]. Agile teams aim to work closely with the customer for ensuring they are not forcing solutions on the customer. This is also the reason why the fourth principle of the Agile Manifesto reminds everyone: “Business people and developers must work together daily throughout the project” [2].

3.4 Think Compass, Not Map

This is the understanding that although a roadmap would help, in a complex environment it is more important to have a compass for navigation because any kind of plan will change during the journey [27].

In general, the values and principles of the Agile Manifesto also serve more like a compass than a map. Not only does the Agile Manifesto explicitly value “Responding to change over following a plan,” the second principle also says “Welcome changing requirements, even late in development. Agile processes harness change for the customer’s competitive advantage” [2]. This acknowledges that change will happen frequently and will require (continuous) re-planning. Agile is known for addressing large and complex problems in an iterative manner. The third principle of the Agile Manifesto states, “Deliver working software frequently, from a couple of weeks to a couple of months, with a preference to the shorter timescale” [2]. With Agile, frequent delivery enables learning and re-calibrates the focus on what is requested next.

For addressing complexity, Agile practitioners apply various models like Cynefin, the work of Ralph Stacey, or Human Systems Dynamics [13, 16, 25]. Stacey’s recent work emphasizes the importance of reflexive inquiry, which is reflecting on how we are thinking. Adaptive Action, as part of Human Systems Dynamics (HSD), is an approach to planning that addresses any complex issue [1, 11]. The approach guides in repeatedly asking the following three questions:

- What? This question is asked to understand the current situation. In this sense, it gathers data about what is happening and what problems need to be addressed.
- So what? The second question is asked to learn what the data just gathered really means. This is about generating insights.
- Now what? The final question is to decide what could be done as the next step to influence change in the system. The next action is decided by answering this question.

Having answered all three questions and having taken the action that was decided, the next round of asking and answering the three questions begins. Since every action taken within a Complex Adaptive System has the potential to shift the situation in unpredictable—but observable—ways, new data must be gathered from that actual state of affairs. This is the idea underlying the Agile practice of frequent deliveries. Developers can quickly correct the course of development with frequent deliveries in uncertain changing situations. The Adaptive Action cycle is also built into Agile retrospectives.

A retrospective opens by setting the stage, before gathering data (that is asking “What?”), followed by generating insights (that is asking “So what?”), and finally deciding what to do (that is asking “Now what?”) before closing the retrospective till the next iteration [7]. In this sense, every retrospective is an Adaptive Action cycle.

3.5 Examples

Much of the literature on creating Theory of Change focuses on bottom-up, step-by-step approaches. This can obscure the dynamic nature of change in Complex Adaptive Systems, and lead people to treat Theory of Change as a map, not a compass [12]. There is no single way to create a Theory of Change, or rather there are multiple ways to create one. Organizations often begin to work with Theory of Change using a triangle template that was developed by funders for evaluating funded programs (see Fig. 1) [6].

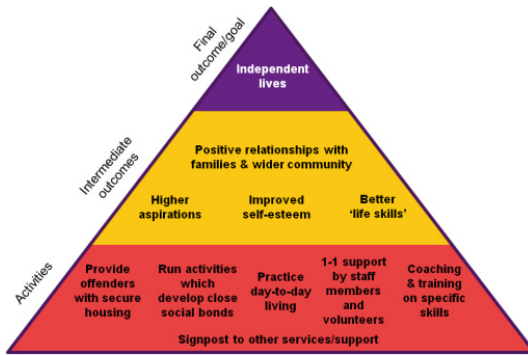


Fig. 1. CES Planning Triangle© presentation of a Theory of Change (for a supported housing project) [6]

“The triangle represents hierarchy” [12] and can also imply judgment that can be detrimental to achieving the hoped-for change. Cara Turner, CEO of Project codeX, emphasized in a private conversation that “when the triangle model is used as a basis for evaluation, the current state forms the ‘base’ of the triangle and, it implies that a steady linear progression upwards is needed to reach the single most desirable and seemingly superior outcome. [...] This could be especially detrimental when equity and social equality are at stake as part of the Theory of Change.”

Another way of developing and representing a Theory of Change uses a “Logic Model” [30], Yet, by themselves, both the triangle and the Logic Model fail to account for new learning about underlying assumptions. They “[a]re not good at showing the dynamic features of a project. They create the impression that inputs and activities happen first, followed neatly by outcomes; in most projects, different aspects occur at different points in time” [12]. Instead of a bottom-up view, Turner extended the CES Planning Triangle by drawing a continuous loop cycle to refine and represent the codeX Theory of Change (see Fig. 2) [22]. Implementing this continuous loop takes into account that, “[m]ost logic models show ‘one pass’ through the intervention, but many interventions

depend on activating a ‘virtuous circle’ where an initial success creates the conditions for further success. This means that evaluation needs to get early evidence of these small changes, and track changes throughout implementation.” [23].

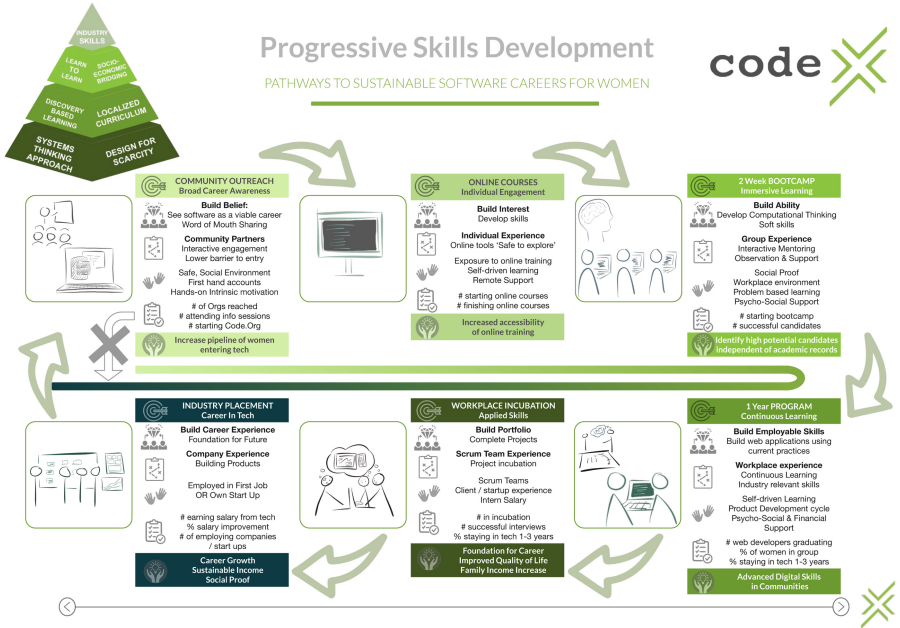


Fig. 2. Presenting the Project codeX Theory of change with a continuous loop cycle [22]

4 Discussion and Conclusion

The Innovation Network regularly surveys non-profit organizations [14]. In their last report, they discovered that 58% of non-profit organizations have a Theory of Change [26]. Unfortunately, from this report, it is unclear how many organizations only created their Theory of Change once versus how many are using it and revising it throughout implementation. Certainly, a Theory of Change will not enable transformation if it’s only used at the beginning of an initiative to secure funding and launch. Nor will it enable transformation if it is only consulted, without active adaptation, to evaluate the initiative’s success or failure at potential endpoints [27]. Like an Agile Charter, Theory of Change is about (a) developing a joint understanding of an endeavor, and it is about (b) continuing to adapt and respond to change as more is learned. Or as Valters puts it, “[p]erhaps the greatest contribution of Theory of Change will lie in helping carve out a space for genuine critical reflection [...]” [27].

Further research is warranted to learn about additional beneficial connections between Agile practice and the creation and continuous refinement of Theories of

Change. In this paper, we have explained the Theory of Change, one of the fundamental approaches frequently used for planning, evaluating, and guiding change in the sustainability sector. By examining Valters' four principles of Theory of Change -Focus on the process, Prioritize learning, Be locally-led, and Think compass, not map -we created a connection to agile values, principles, and practices. For better understanding, we provided some examples of Theory of Change.

As Theory of Change is fundamental to many organizations and groups working for Sustainable Development goals, Agilists need to understand it, if they will be supporting these entities in the sustainability sector. This paper has connected Theory of Change with agile mindsets and methods (values, principles, and practices) so Theory of Change can also serve as a foundation for agilists especially when practicing sustainability by Agile in the sustainability sector.

We hope with this new understanding that agile practitioners will be able to more effectively promote sustainability by Agile. We encourage practitioners to offer support to NGOs and other organizations to continue learning about the connections between Agile and Theory of Change in working for a sustainable future.

References

1. Adaptive Action. <https://www.hsdinstitute.org/resources/adaptive-action.html>. Accessed 14 Mar 2022
2. Agile Manifesto (2001). <http://agilemanifesto.org/>. Accessed 18 Mar 2022
3. Argyris, C., Schön, D.A.: *Organizational Learning: A Theory of Action Perspective*. Addison-Wesley Publishing Company, Reading, Mass (1978)
4. Beck, K.: *EXtreme Programming eXplained: Embrace Change*. Addison Wesley, Reading, Mass (2000)
5. Center for Theory of Change. <https://www.theoryofchange.org/>. Accessed 18 Mar 2022
6. CES Planning Triangle®. <https://www.thinknpc.org/wp-content/uploads/2018/07/Creating-your-theory-of-change1.pdf>. Accessed 28 Mar 2022
7. Derby, E., Larsen, D.: *Agile Retrospectives: Making Good Teams Great*. Pragmatic Programmers LLC (2006)
8. Eckstein, J., Buck, J.: *Company-Wide Agility with Beyond Budgeting, Open Space & Sociocracy: Survive & thrive on Disruption*. Eckstein, Braunschweig, Germany (2020)
9. Eckstein, J.: *Retrospectives for Organizational Change. An Agile Approach*, 2nd edn. Eckstein, Braunschweig, Germany (2019)
10. Funnell, S., Rogers, P.: *Purposeful Program Theory: Effective Use of Theories of Change and Logic Models*. Jossey Bass, San Francisco, CA (2011)
11. Eoyang, G., Holladay, R.: *Adaptive Action: Leveraging Uncertainty in Your Organization*. Stanford University Press, Stanford, CA. Kindle Edition (2013)
12. Harries, E., Hodgson, L., Noble, J.: *Creating Your Theory of Change*. New Philanthropy Capital, London, UK (2014). <https://golab.bsg.ox.ac.uk/knowledge-bank/resources/creating-your-theory-change-npcs-practical-guide/>. Accessed 28 Mar 2022
13. HSD: Human Systems Dynamics. <https://www.hsdinstitute.org/>. Accessed 23 Mar 2022
14. Innovation Network: <https://www.innonet.org/>. Accessed 23 Mar 2022
15. Kua, P.: *The Retrospective Handbook: A Guide for Agile Teams* (2013)
16. Kurtz, C.F., Snowden, D.: The new dynamics of strategy: sense-making in a complex and complicated world. *IBM Syst. J.* **42**(3), 462–483 (2003). <http://tinyurl.com/kurtz-pdf>. Accessed 23 Mar 2022

17. Larsen, D., Nies, A.: *Liftoff: Start and Sustain Successful Agile Teams*. 2nd Edition Pragmatic Bookshelf (2016)
18. Mezick, D., et al.: *The OpenSpace Agility Handbook*, 2nd edn. Freestanding Press (2015)
19. Owen, H.: *Open Space Technology. A User's Guide*, 3rd ed. Berrett-Koehler Publishers Inc. San Francisco, USA (2008)
20. Owen, H.: *Wave Rider: Leadership for High Performance in a Self-Organizing World*. Berrett-Koehler Publishers Inc., San Francisco, USA (2008)
21. Patton, J., Economy, P.: *User Story Mapping: Discover the Whole Story, Build the Right Product*. O'Reilly Media (2014)
22. Project codex. https://www.projectcodex.org/docs/ProjectcodeX_Theory_of_Change.pdf
23. Rogers, P.: *Using Programme Theory to Evaluate Complicated and Complex Aspects of Interventions*, vol. 14, no. 1, pp. 29 – 48. SAGE Publications, Los Angeles (2008). <https://doi.org/10.1177/1356389007084674>. <https://journals.sagepub.com/doi/10.1177/1356389007084674>. Accessed 28 Mar 2022
24. Rose, S., Wynne, M., Hellesøy, A.: *The Cucumber for Java Book. Behaviour-Driven Development for Testers and Developers*. The Pragmatic Bookshelf, LLC. Kindle Edition (2015)
25. Stacey, R.D.: *Strategic Management and Organizational Dynamics*, 6th edn. Pearson Education Ltd., Harlow, UK (2011)
26. State of Evaluation. <https://www.innonet.org/news-insights/resources/state-of-evaluation-2016-evaluation-capacity-and-practice-in-the-nonprofit-sector/>. Accessed 23 Mar 2022
27. Valters, C.: *Theories of Change: Time for a Radical Approach to Learning in Development*. ODI, London (2015)
28. Vogel, I.: Review of the use of 'Theory of Change' in international development. Review report (2012). https://www.theoryofchange.org/pdf/DFID_ToC_Review_VogelV7.pdf. Accessed 4 Apr 2022
29. Weiss, C.: Nothing as practical as good theory: exploring theory-based evaluation for comprehensive community initiatives for children and families. In: Connell, J.P., Kubisch, A.C., Schorr, L.B., Weiss, C.H. (eds.) *New Approaches to Evaluating Community Initiatives*. Aspen Institute, Washington, DC (1995)
30. World Clean-Up Day. https://global-uploads.webflow.com/60ae9a8dbba8f536b22321fb/612cc7ed1d5b9effcaf52266_Theory%20of%20Change.pdf. Accessed 2022/03/25
31. Zuill, W.: *Mob Programming - A Whole Team Approach* by Woody Zuill (2015). <https://www.agilealliance.org/resources/experience-reports/mob-programming-agile2014/>. Accessed 28 Mar 2022

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.





Enhancing Agile Software Development Sustainability Through the Integration of User Experience and Gamification

Manal Alhammad¹  and Ana Moreno² 

¹ King Saud University, Riyadh, Saudi Arabia
manalhammad@ksu.edu.sa

² Universidad Politécnica de Madrid, Madrid, Spain
ammoreno@fi.upm.es

Abstract. This article provides a rich discussion on how the sustainability of agile development processes can be enhanced. In particular, we focus on a recently developed framework, named GLUX, that integrates Lean UX into Scrum. GLUX's main goal is to facilitate a seamless integration between agile and user experience (UX) by using gamification to motivate agile teams to adopt a user-centered mindset and carry out UX activities collaboratively throughout the development process. Our role as software researchers is to contribute towards improving software sustainability and provide the software engineering community with the tools and techniques that will improve the human, economic, and environmental sustainability of software development. We found that GLUX addresses human sustainability by empowering self-sufficient, problem-focused teams, building a motivating and engaging environment, and developing team cooperation. Economic sustainability is addressed by minimizing UX debt and using gamification techniques to direct the focus of the behavior and mindset of agile teams towards value creation. Finally, environmental sustainability is promoted by encouraging agile teams to build a minimum viable product (MVP).

Keywords: Software sustainability · Agile · Lean UX · Gamification

1 Introduction

There is no question nowadays that software has a major impact on sustainability [1]. As Calero et al. in [2] state, software can be considered as part of the problem, but also has a lot to do with the solution. These same authors present a literature review of how sustainability is addressed in software development [3]. They identify two different perspectives. On one side, they look at what is referred to as software sustainability (SOS), which is concerned with how to make software production more sustainable, covering everything from the people, through the process and the business, to the product. On the other hand, some literature considers software as part of sustainability (SAPOS), where software is considered as a new dimension of this attribute.

Sustainable software engineering (SSE) is an emerging discipline that aims to address the long-term impact of designing, building, deploying, and maintaining software products [4]. In this vein, Calero and Piattini identify three dimensions of software sustainability, in line with the standard definition of sustainability, based on the three types of resources (human, economic, energy) essential in the software life cycle processes [3]. Human sustainability refers to analyzing and addressing the impact of software development on the sociological and psychological aspects of the software engineering community and its individuals. Economic sustainability refers to the means by which the software process protects stakeholder investments, ensures benefits, reduces risks, and maintains assets. Environmental sustainability refers to the impact of software development and maintenance on energy consumption and the usage of other resources.

Sustainability has also been recently addressed in Agile software development. For example, in [5], Eckstein and Melo analyze how sustainability is promoted by the agile philosophy, and give a detailed discussion about how it is considered in each of the agile principles. Obradović, Todorović, and Bushuyev in [6] discuss the relation between agile project management and sustainability and conclude that they are overlapping and that agile project management requires the implementation of sustainability aspects. Ochoa-Zambrano in [7] examines how collective intelligence fits into the agile manifesto and its values and principles and discusses the role of collective intelligence as a tool to enhance agile and sustainability by emphasizing team collaboration and learning. Additionally, there are a few attempts to modify the agile development process to incorporate practices that contribute to making more sustainable software products [8].

In this paper, we look into how user experience (UX) and gamification can reinforce the human, economic, and environmental sustainability of the agile development process. Through the lens of SOS, we particularly examine the recently developed GLUX (Gamified Lean UX) framework whose main goal is to facilitate a seamless integration between agile and UX by using gamification to motivate agile teams to adopt a user-centered mindset and carry out UX activities collaboratively throughout the development process.

2 What is GLUX?

Endeavors to combine agile and UX are based on the premise that both disciplines share common principles, such as iterative development, emphasis on the user, and team coherence [9]. Even though the potential benefits of this integration are recognized, existing approaches have been found to have a number of limitations [10]. For example, the process is not fully integrated [11], agile developers do not have a UX mindset [12], or there is a lack of rigorous empirical studies [9].

In order to address some of the challenges of integrating agile and UX, we have developed the GLUX framework [13]. GLUX engages practitioners to collaboratively integrate Lean UX activities within Scrum. Lean UX is a lightweight and iterative UX design process that is based on design thinking, lean startup and Agile [14]. GLUX takes into account the different characteristics and complexity of software engineer and agile team motivational factors. It is aimed first and foremost for small Scrum teams who are struggling with integrating UX activities into the development process, particularly in

the absence of UX specialists. Even if a UX specialist is on hand, the GLUX framework can help align the workflow of the whole team towards building user-centered software projects through a shared understanding. As illustrated in Fig. 1, the GLUX framework is divided into two fundamental parts:

- Five Lean UX tactics for integration into Scrum: Hypotheses, Design Studio, Experiment Stories, Minimum Viable Product (MVP), and Weekly User Experiments.
- A customizable gamification strategy based on three game techniques: rewards (points and badges), challenges, and levels.

The full documentation of the GLUX framework is available at <https://bit.ly/2Xp1H1A>.

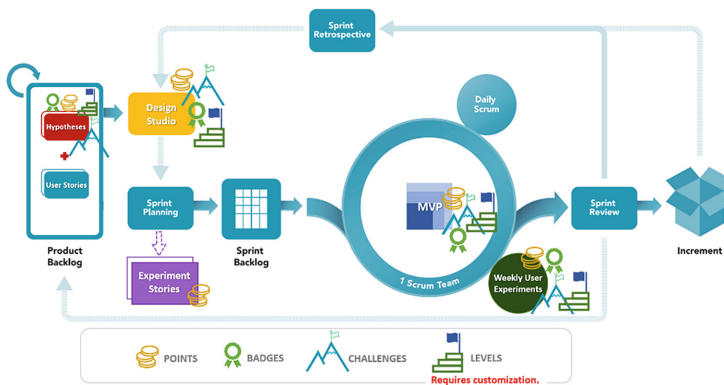


Fig. 1. A general overview of the GLUX framework, integrating Scrum with Lean UX using gamification.

A typical scenario of an Agile team following GLUX is as follows: the team creates a list of hypotheses on their assumptions about the needs of potential users, which they discuss with the product owner (PO) during product backlog refinement. The sprint kicks off with the design studio, where the team picks a hypothesis as a theme to guide the work of the sprint and start sketching and discussing design ideas accordingly. The sprint planning meeting should take place immediately after the design studio. Apart from deciding the user stories to be developed, the team also plans for the weekly user experiment during sprint planning, in which the team puts a version of their developing product, i.e., an MVP, to the test in order to gather feedback from the user. User experiment plans are captured in what is called an experiment story.

The team is rewarded for applying each Lean UX tactic and receives additional rewards for doing it collaboratively. The Scrum team is also encouraged by special rewards for employing Lean UX tactics for the first time, as well as for addressing some Lean UX challenges. Challenges are typically set by the team based on current UX-related difficulties and issues.

3 How Does GLUX Address the Three Dimensions of SOS?

GLUX was applied for two academic semesters into a novel graduate software engineering course offered as part of the MS in Software Engineering program at the Universidad Politécnica de Madrid. The discussion below is based on the lessons learned and the preliminary evidence that we got from this experience reported in [19].

3.1 Human Sustainability

Human sustainability deals with sociological and psychological aspects of software development and developers [3]. Such aspects are critical in an intellectual activity like software construction [15].

In this sense, as discussed in the previous section, one of the main challenges to Agile UX is that developers do not have a UX mindset [13]. This challenge means that agile teams continuously prioritize delivering fast and “working” features, while paying less attention to UX aspects. This tendency mostly emerges from four key issues. First, the Agile Manifesto term “working software” came over time to be misinterpreted, and agile teams began to shift their focus to delivering functioning software rather than to supplying valuable software. Second, due to the lack of knowledge on and training in UX design, software engineers have very little motivation for and interest in UX work. Third, it is argued that “resistance to change” is linked to the reluctance to integrate UX practices into agile development. The last issue is related to what is called the curse of knowledge in which agile teams may come to believe that they know better than the user what features to build and how they should be designed and delivered.

In GLUX, a UX mindset is explicitly promoted and rewarded in four ways:

Empowering Teams to Become Self-Sufficient. GLUX aims to empower agile teams with the skills and mindset needed to facilitate the integration of UX into agile through a cohesive process (Fig. 1) vs. a parallel or dual track. This would ultimately help the team to consider the Agile-UX process as a single collaborative process in which UX work is part of the agile development process.

Establishing a Problem-Focused Team. Rather than providing a set of features for implementation, GLUX guides agile teams to strive for continual improvement and builds trust within teams which take part in designing the solutions that can help to address a business outcome or a user need in the form of hypotheses, design sketches, or MVPs. This can give teams a greater sense of pride, control, and ownership over the solutions they came up with.

Building a Motivating and Engaging Environment. As discussed earlier, Agile UX faces to different challenges. Darin et al. in [16] found that engaging the development team in the user-centered design (UCD) process would help make the integration a more natural process. They also emphasized the importance of keeping the team motivated and committed. Some agile teams need extrinsic motivators to perform UX activities. The GLUX framework aims to support agile teams in being more proactive about UX work through its gamification strategy. Gamification in SE is seen as a promising technique to improve software engineers’ motivation and engagement and to promote SE

best practices [17]. The gamification strategy was designed based on a comprehensive analysis of the nature and characteristics of agile teams [13]. For instance, we found that one of the key motivators of agile teams is having a sense of achievement and a regular stream of feedback on the work they do [18]. As a result, GLUX’s gamification strategy includes a rewards system, a challenges system, and a levels system. The rewards system provides the team with a recognition of their achievements. The challenges system provides the team with a sense of achievement and encourages the entire team to be more goal-driven, focused, and collaborative. The levels system provides the team with a sense of progression, which can consequently improve the team’s motivation to collaborate and apply Lean UX tactics. Furthermore, GLUX’s gamification strategy establishes a fun environment t using elements such as badges and an achievements board. Figure 2 shows also a few examples of the cheat cards we created for the master’s course¹ where we applied GLUX [19]. Cheat cards are essentially a visual summary of each Lean UX tactic in terms of the rules for integrating such tactics in Scrum, how the scoring system works, and one proposed challenge.

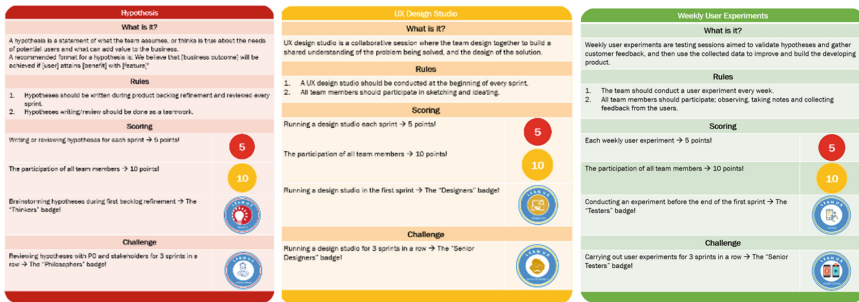


Fig. 2. Three examples of GLUX Cheat Cards, providing a visual summary of GLUX’s rules and scoring system.

Developing a Cooperative Team. GLUX’s gamification strategy is team based, where the whole team is encouraged to participate in some activities that require collaboration for the team to earn the associated rewards. Rewarding the collaboration of the entire team is aimed at facilitating a frequent and quick way to exchange ideas, allowing the team to move forward quickly into the right direction with everyone having a clear picture of the envisioned increment in each sprint. In addition, collaboration among team members from different backgrounds and expertise can help the team to learn from each other (for example, during the design studio developers can understand some aspects of the proposed design or suggest some changes from their perspective, and designers can understand the technical complexity of implementing a particular design). Ultimately, collaboration, experience sharing, and learning from each other can enhance the work performed by individuals in the future and encourage them to pass on the knowledge and skills they learned to other teams.

¹ <https://bit.ly/GLUXGUIDE2>.

3.2 Economic Sustainability

Economic sustainability is either explicitly or implicitly promoted in some of the 12 agile development principles [5]. For example, the eighth principle, which reads “Agile processes promote sustainable development. The sponsors, developers, and users should be able to maintain a constant pace indefinitely”, promotes economic sustainability explicitly by advocating the development of valuable features and avoiding what is known as feature creep. On the other hand, agile development promotes economic sustainability implicitly by enabling the team to measure the economic impact of the delivered product and constantly learn about and improve both the product and the process.

GLUX addresses economic sustainability by **minimizing UX debt** to reduce rework. If UX issues are not given as much attention as functional issues, it will be all too easy for UX debt to creep in and pile up. UX debt result in teams building up UX deficiencies and then being forced to carry out additional rework. This phenomenon would eventually have a bearing on key performance indicators, as it would also impact developer productivity. More UX debt means extra hours of work at greater expense for the business. While a few other factors may lead to additional working hours, the causes can usually be traced back to the failure to focus on continuous quality improvement [20]. In GLUX, UX debt is addressed through a gamification technique called sprint challenge. The sprint challenge encourages prompt and collaborative proactivity with respect to UX debt issues. It starts with the team identifying the challenging process issues related to UX that they face and which they have listed from easiest to hardest. Starting with the easiest, the team picks a challenge every other sprint, sets it as the sprint challenge and works towards addressing this question. A sprint challenge keeps the team more focused on the pending UX issues that may turn into UX debt and result in rework. This way, the team is able to respond rapidly and at a much lower cost. It also helps teams to be more proactive about new technologies or new opportunities in any form.

Economic sustainability is also reinforced in GLUX mainly through the **hypothesis definition**. By creating a hypothesis, agile teams map every feature they intend to develop to a business outcome and a user benefit. Any feature or task that does not contribute to achieving or improving a business outcome or a user benefit is considered unnecessary. The more unnecessary features the team can avoid, the less resources are wasted. Additionally, Lean UX offers a canvas to prioritize hypotheses based on risk and value [21]. Risk refers to how damaging it would be for the business or product if the team was wrong about this hypothesis. Value refers to the perceived value that will be generated from developing the feature. Based on this prioritization, the team should only spend their resources on testing hypotheses with high risk and high value.

3.3 Environmental Sustainability

According to Calero et al. in [22], the key to software sustainability is to improve its power consumption. However, software, unlike hardware, production and usage has not witnessed continuous advancements in terms of energy efficiency [23]. GLUX addresses environmental sustainability in two ways. First, by **building just enough product** to validate the hypothesis. Before jumping into building a complete feature or increment, agile

teams are encouraged to build an MVP, that is, the simplest version of the envisioned feature. The MVP is then validated with real users to check whether the feature provides value to the user or the business. If the feature is found to be useful, the team works on improving that feature and releases it to the end user. If not, the team can roll back that feature with a minimal environmental cost. Building a hypothesis-driven MVP can minimize energy waste by building the smallest piece of software to validate the usefulness of that feature. Second, GLUX addresses environmental sustainability by promoting collaboration in-between developers, designers, product owners, and project managers on most activities. With increased **cross-functional and continuous collaboration** among the whole team, fewer resources are used for documentation and correspondence.

4 Conclusion

In this paper, we examined how the integration of Lean UX and gamification can enhance the sustainability of agile development processes from the human, economic, and environmental perspectives. We presented a recently developed framework called GLUX. Through gamification, GLUX aims to engage agile teams in integrating Lean UX tactics into Scrum in a collaborative way. We have seen how human sustainability can be addressed in GLUX by empowering self-sufficient teams, establishing problem-focused teams, building a motivating and engaging environment, and developing a cooperative team. We have also discussed how GLUX addresses economic sustainability by minimizing UX debt and using gamification techniques to influence the behavior and mindset of agile teams to focus on creating value and reducing waste. Finally, we looked into how GLUX promotes environmental sustainability by encouraging agile teams to build just enough product to validate their hypothesis and continuously collaborating on most activities throughout the development process.

We acknowledge that there are not enough efficient and validated techniques and tools to enhance software sustainability. We have used GLUX in an academic setting with promising results [19]. However, it is fundamental to empirically validate how GLUX can contribute to improving the sustainability of software development in a real-world context. Partnering with agile teams from industry would be useful to verify the feasibility of the proposed ideas.

It is also important in this context to discuss how highly usable software products might have the opposite effect on sustainability. Simple and effective software might in fact encourage people to use it more often and get more people to use it as well without a specific purpose. This paradox is related to the so-called rebound effect, which is generally defined as “the difference between the expected and the actual environmental savings from efficiency improvements” [24]. Promoting the responsible consumption of software products can lead to a win-win situation, where users are more aware and engaged as active participants in sustainability practices.

References

1. Belkhir, L., Elmeligi, A.: Assessing ICT global emissions footprint: trends to 2040 & recommendations. *J. Clean. Prod.* **177**, 448–463 (2018)
2. Calero, C., Angeles Moraga, M., Garcia, F.: Software, sustainability, and UN sustainable development goals. *IT Prof.* **24**(1), 41–48 (2022)
3. Calero, C., Piattini, M.: Puzzling out software sustainability. *Sustain. Comput. Informatics Syst.* **16**, 117–124 (2017)
4. M. (Microsoft) Fabrizio: Sustainable Software Engineering (SSE) and the role and responsibilities of a Sustainable Software Engineer. Microsoft - Sustainable Software (2022). <https://devblogs.microsoft.com/sustainable-software/sustainable-software-engineering-sse-and-the-role-and-responsibilities-of-a-sustainable-software-engineer/>. Accessed 07 Apr 2022
5. Eckstein, J., Melo, C.deO.: Sustainability: delivering agility's promise. *Softw. Sustain.*, 215–241 (2021)
6. Obradović, V., Todorović, M., Bushuyev, S.: Sustainability and agility in project management: contradictory or complementary? In: Shakhovska, N., Medykovskyy, M.O. (eds.) CSIT 2018. AISC, vol. 871, pp. 522–532. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-01069-0_37
7. Ochoa-Zambrano, J.: How collective intelligence can gear agility with sustainability. In: Gregory, P., Kruchten, P. (eds.) *Agile Processes in Software Engineering and Extreme Programming – Workshops. LNBP*, vol. 426, pp. 69–77. Springer, Cham (2021). https://doi.org/10.1007/978-3-030-88583-0_7
8. Dick, M., Drangmeister, J., Kern, E., Naumann, S.: Green software engineering with agile methods. In: 2013 2nd International Workshop on Green and Sustainable Software, GREENS 2013 - Proc., pp. 78–85 (2013)
9. Da Silva, T.S., Silveira, M.S., Maurer, F., Silveira, F.F.: The evolution of agile UXD. *Inf. Softw. Technol.* **102**, 1–5 (2018)
10. Curcio, K., Santana, R., Reinehr, S., Malucelli, A.: Usability in agile software development: a tertiary study. *Comput. Stand. Interfaces* **64**, 61–77 (2019)
11. Kashfi, P., Feldt, R., Nilsson, A.: Integrating UX principles and practices into software development organizations: a case study of influencing events. *J. Syst. Softw.* **154**, 37–58 (2019)
12. Ananjeva, A., Persson, J.S., Bruun, A.: Integrating UX work with agile development through user stories: an action research study in a small software company. *J. Syst. Softw.* **170**, 110785 (2020)
13. Alhammad, M.M.: A gamified framework to integrate user experience into agile software development process. *ETSI Informatica* (2020)
14. Gothelf, J., Seiden, J.: *Lean ux*. O'Reilly Media, Inc., Sebastopol (2021)
15. Barroso, A.S., Madureira, J.S., Soares, M.S., do Nascimento, R.P.C.: Influence of human personality in software engineering-a systematic literature review. In: *International Conference on Enterprise Information Systems*, vol. 2, pp. 53–62 (2017)
16. Darin, T., Castro, R., Carneiro, N., Almeida, R., Andrade, R.: Integrating HCI perspective into a mobile software development team: strategies and lessons from the field. In: *Proceedings of the 21st International Conference on Human-Computer Interaction with Mobile Devices and Services, MobileHCI 2019* (2019)
17. Pedreira, O., García, F., Brisaboa, N., Piattini, M.: Gamification in software engineering: a systematic mapping. *Inf. Softw. Technol.* **57**, 157–168 (2015)
18. Beecham, S., Noll, J.: What motivates software engineers working in global software development? In: Abrahamsson, P., Corral, L., Oivo, M., Russo, B. (eds.) *PROFES 2015. LNCS*, vol. 9459, pp. 193–209. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-26844-6_14

19. Alhammad, M., Moreno, A.: Integrating user experience into agile: an experience report on lean UX and scrum. In: 44th International Conference on Software Engineering: Software Engineering Education and Training (ICSE-SEET 2022), p. 12 (2022)
20. Ganguly, V.R.: Sustainable software development: Code4research. *VOEB-Mitteilungen* **71**(1), 171–180 (2018)
21. Gothelf, J.: The Hypothesis Prioritization Canvas. Jeff Gothelf (2019). <https://jeffgothelf.com/blog/the-hypothesis-prioritization-canvas/>. Accessed 07 Apr 2022
22. Calero, C., Moraga, M.Á., Piattini, M.: Introduction to software sustainability BT - software sustainability. In: Calero, C., Moraga, M.Á., Piattini, M. (eds.) *Software Sustainability*, pp. 1–15. Springer, Cham (2021)
23. Capra, E., Francalanci, C., Slaughter, S.A.: Is software ‘green’? Application development environments and energy efficiency in open source applications. *Inf. Softw. Technol.* **54**(1), 60–71 (2012)
24. Vivanco, D.F., McDowall, W., Freire-González, J., Kemp, R., van der Voet, E.: The foundations of the environmental rebound effect and its contribution towards a general framework. *Ecol. Econ.* **125**, 60–69 (2016)


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.





Sustainable IT in an Agile DevOps Setup Leads to a Shift Left in Sustainability Engineering

Alexander Poth^(✉) , Daniela Eißfeldt, Christian Heimann, and Stefan Waschk

Volkswagen AG, Berliner Ring 2, 38436 Wolfsburg, Germany
{alexander.poth,daniela.eissfeldt,christian.heimann,
stefan.waschk}@volkswagen.de

Abstract. Today green IT is mostly driven by the measurement of CO₂e of data centers. However, this is a symptom treatment approach, since the operating parameters of software are defined during build-time. This implies that the consumption during run-time of a software cannot be changed in a wide range. To ensure that enterprise IT can be operated within a higher sustainable setup the software and systems engineering has to consider sustainability aspects during development phase. Furthermore, sustainability is more than measuring and optimizing CO₂e of applications – it includes e.g. reuse aspects. Each software component which is reused reduces resource allocation during development and maintenance. IT sustainability step by step becomes a quality characteristic of software. This work presents a more holistic view for sustainable software engineering from an enterprise IT perspective which can be integrated into agile software development especially within DevOps teams.

Keywords: sustainable software engineering · green coding · agile transformation · green IT · ISO 14001 · DevOps

1 Motivation, Context and Methodology

Many enterprises in different countries and regions are working hard to become more sustainable like [1, 2, 3]. A typical starting point is to measure and optimize respective CO₂e emissions. To ensure professional management of the optimization standards like the ISO 14001 are used [4]. However, many companies are focused on their production or operations while measuring and improving their sustainability footprint. With a focus on enterprise IT the situation becomes more complex as to use an operation service driven approach, because the parameters for IT systems and software operations are defined in an earlier life-cycle stage. The important life-cycle stage to ensure a sustainable software of IT systems is defined during design and development. To increase the effectiveness of the sustainability actions and efforts an interdisciplinary agile team – with skills in design, development and operation - can facilitate a shift left of sustainability topics into the build phase of enterprise IT software. This shift left approach has to be aligned with the enterprise environmental and sustainability strategy and adapted to the working level procedures. For agile teams this implies to aware of sustainable software engineering

© The Author(s) 2024

P. Kruchten and P. Gregory (Eds.): XP 2022/2023 Workshops, LNBP 489, pp. 21–28, 2024.

https://doi.org/10.1007/978-3-031-48550-3_3

with the corresponding methods and tools fostering the integration of sustainable engineering into their value stream workflows. The approach has to consider aspects from the Greensoft model [5], impacts of SLAs [6] for IT based/supported services [7] and the “bin packing” problem [8] to ensure that resources are adequate allocated for a high utilization.

The research questions around this enterprise IT sustainability setup are:

RQ1: What are the main dimensions for a holistic sustainable software engineering?

RQ2: What is needed for agile teams to perform sustainable software engineering?

Section 2 elaborates the sustainability model for enterprise IT. Section 3 presents the use case for agile DevOps team and Sect. 4 gives an example form the Volkswagen Group IT. Finally, Sect. 5 concludes by summarizing the article’s key contributions to research and practice and giving an outlook to the authors’ ongoing research activities.

2 Methodology and Outcome Design

An Action Research (AR) [9] approach is used to ensure that the outputs are usable by the value stream teams and the outcomes fit into real enterprise IT setups. To answer RQ1 the sustainability model is derived and refined as followed: To establish a shift left in sustainability engineering a life-cycle approach is developed. Furthermore, a consistent refinement from the overall sustainability objective to the product or service specific actions has to be established. To address this two dimensions the matrix of Fig. 1 was developed. Each dimension is based on factors. One dimension (the y-axis in the figure) represents the organizational abstraction level, the other dimension represents the life-cycle phases (the x-axis in the figure) of the software.

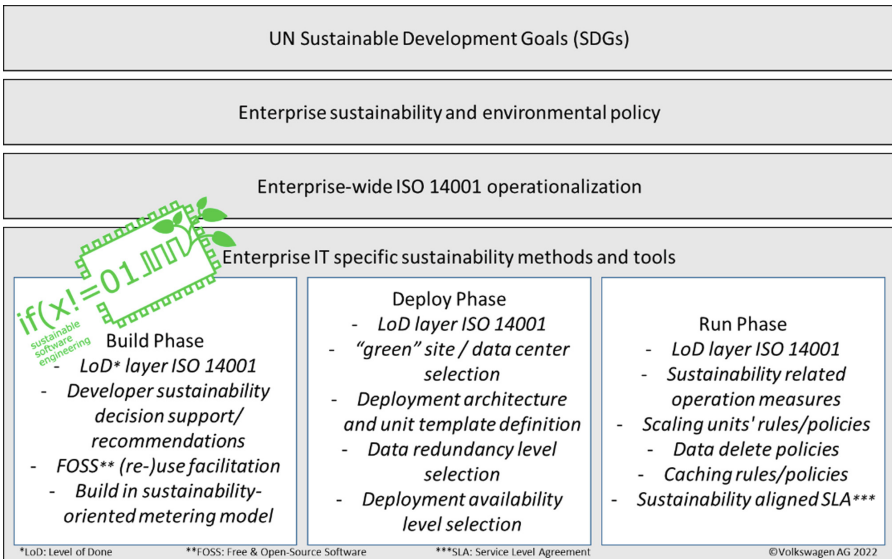


Fig. 1. Sustainability model for enterprise IT software and systems.

The figure has on its y-axis four levels respectively factors.

The *UN SDG* [10] – UN Sustainable Development Goals are the most generic sustainability objectives. These are a good starting point for all sustainability initiatives.

The *enterprise sustainability and environmental policy* selects from the UN SDG or can be mapped to them. The policies set focus and foster transparency in the effects of activities.

The *enterprise wide ISO 14001 operationalization* is to refine respective instantiate and operate the enterprise policy. The management system establishes an efficient organization of environmental improvement. Important is that this is the base for a deduction and refinement to specific business domains and value streams. It sets the boundaries in which strategies are acted. Typical strategies are efficiency, consistency and sufficiency in the context of sustainability [11].

For the *enterprise IT specific sustainability methods and tools* an IT domain specific set of methods and tools for implementation is used. The set is used to facilitate a wide adoption by agile value stream teams. This methods address sustainability related service pricing, effective reuse e.g. based on FOSS (Free and Open Source Software) and systematic sustainable software engineering recommendations.

The x-axis is structured into three core phases. Within the phases are specific factors which have to be considered by the engineers. The core phases are applied to releases of the software.

The *build phase* defines all the parameters for the following phases. Examples include the modularization of components which are workload-sensitive to be able to scale them elastic to the current workload. Implement algorithms which are resource efficient for the workload. Offer parameterization of the software to address specific data life-time or redundancy of the data and software components (availability).

The *deployment phase* determinates the operation environment and its “green-factor” for the following phase. Here it is important to select the most energy efficient environment available parameters. Two advices are to select the data center with a highest Power Usage Effectiveness (PUE) [12] available, and to select the most energy efficient architecture platform like ARM over AMD over Intel CPUs. This selection is motivated by Koomey’s Law [13] which indicates a continuous efficiency optimization with each CPU generation. However, here the options can be limited by the delivered software components which are for e.g. build for x86 or enterprise policies which includes a commitment to x86-architecture. Furthermore, to select adequate data-lifetime policies and redundancy strategies is recommended.

The *run phase* optimize the resource consumption within the determined environment and pre-defined parameters. Examples are to select the newest generation of instances for the most power efficient execution, use optimized instance types for the workloads, adjust scaling policies optimize caching and routing for the specific software system. Furthermore, establishing sustainability related measure as base for further optimization is advised.

A decommission phase is not defined as a core phase, because with a good architecture combined with continuous refactoring keeps the software “young”. However, every

software has its end of life which also addresses the switch or migration to another software which it is not in scope in our presented view from the sustainability perspective. Related data has to be handled with an objective to delete or reuse it in another IT system.

Furthermore, a plan phase is not defined, because as IT sustainability is a IT delivery characteristic it is not a topic to elaborate it with the business like the Product Owner. This is similar to IT Security – it is driven by the IT architecture and for its implementation the business is asked for some decisions were needed, but is made and driven by IT experts. The business has to ensure that only valuable and relevant functionality and capabilities are built by the IT as a core contribution of sustainability form a business perspective. The entire IT deliverables have to be seen as a part of the business sustainability concept.

The sustainability model with its dimensions is open for additional enterprise respective product/service domain specific factors. For example in a finance domain established mainframe system based infrastructure FOSS reuse is a limited applicable factor – or reduced to an inner-source mindset. Figure 1 shows a basic set of factors which are generic for many enterprise IT setups. The figure is inspired by the Plan-Build-Run approach [14]. Between Build and Run a Deployment is added to make transparent that here some important setting are made which can have impact to the sustainability of the service delivery. The Build phase includes solution architecture and design. The Plan phase is not focused as it does not address technical IT respective implementation aspects.

To establish all three phases of ISO 14001 teams need to be organized in an agile way like with the LoD layer approach [15]. Fostering common strategic aspects such as metrics or reuse – all which is needed to work over the three phases “smooth” is important as well.

The described sustainability model shows that IT and software sustainability is an additional quality characteristic of deliverables which have to be developed and established in enterprise IT organizations. The sustainability model uses efficiency strategy building blocks like algorithm optimization, consistency strategy building blocks like reuse via FOSS and it uses the sufficiency strategy building blocks like adequate scaling units.

3 Leveraging Sustainability with the Sustainability Model

To address RQ2 depending on the organization, different implementation scenarios are possible. The most flexible setup is found in DevOps value streams. In this cases the initial starting point can be a top-down driven by a values stream team or bottom-up driven by management approach. Furthermore, the run phase can be used as starting point to make a quick-win to optimize within the current predefined parameters footprint driven optimization back to the build phase. Also a build phase initiated sustainability engineering is possible for strategic actions with large levers.

In organizations without DevOps teams an ops driven bottom-up approach is potentially limited by the “silo” borders between ops and dev. A dev driven bottom-up approach has more success, because dev propagates the sustainability options to the later phase within the by design delivery flow. Also a top-down approach which addresses dev and ops has a higher success probability. The higher success is given by the option

to define common sustainability measures as base for e.g. common Objectives & Key Results (OKR) for the dev and ops team in a delivery stream. This ensures that both teams cooperate to achieve the same sustainability objectives.

However, in every point in the proposed sustainability model for enterprise IT is a possible starting point for a sustainability initiative, at least it can optimize the sustainability from a local perspective.

These analysis recommend a rollout scenario depending on the current organizational setup:

- Agile DevOps team: the sustainability initiative can initiate “everywhere”
- Independent agile teams for dev and ops: avoid to start in the ops team, because the silo boarder can limit effective rollout in direction dev team.
- Top-down is always possible by establish alignment of the teams with the defined objectives and goals

The organizational setup of the value stream teams has an influence how effective the sustainability efforts show effects and their impact on the service delivery. With the organizational setup like DevOps teams or management actions like OKR the base for a shift-left of sustainability actions is supported to act sustainable by design were possible.

4 Instantiation and Evaluation

Everything starts with the freedom to act for change. Within the Volkswagen Group IT triggers are established to act on the topic sustainability such as the “1-h project” [16] and initiatives like go2Zero [17]. These triggers encourage teams to invest into their sustainability capabilities to deliver more environmental friendly services to their users. A representative example for an enterprise IT setup is the Test-Runtime execution (T-Rex) cloud testing service DevOps team. The DevOps team of the service applies agile and lean principles and working methods. The team is formed by engineers with a T-shaped skill profile to ensure that all relevant aspects of a cloud-native service are handled like architecture, quality and testing within the software development. Furthermore, the team is not static by design because it is also a training on the job place for vocal education – every 6 months at least a team members either joins or leaves the team. Additionally, the team is composed of internal developers and contractors. The team started its sustainability journey over 2 years ago. Initially it began with explicit actions to optimize the infrastructure footprint during development – a quick win action. A parallel action in direction shift left was to optimize the effects of the software usage. This also includes the evaluation of alternative architecture approaches like serverless [18] and the insight to establish resource allocation related service models – here the shift left journey starts. To professionalize the sustainability actions an alignment with the ISO 14001 followed soon. As the team is using the method kit based efiS® framework [19] with the LoD layer [20] ISO 14001 which was an reusable output from the instantiation. This output is a first component which can be reused by other teams within the Volkswagen Group IT. The deduction of the enterprise environmental policy within IT was an additional outcome and described in [15]. This serves as a blueprint for other teams, too. The refinement work leads to the insight that a wider scaling within an enterprise IT more than the

LoD layer ISO 14001 is needed. Therefore the development of the cheat-sheet for sustainable software engineering was triggered [21] and it will be distributed to interested teams. The cheat-sheet consolidates knowledge about sustainable software engineering for an easy application within the Volkswagen Group IT. Parallel the systematic evaluation of FOSS components was initiated to professionalize software reuse. The gap was that established FOSS evaluations are a manual activity which does not scale for an extensive reuse – the objective was a (semi-)automation of the evaluation. This leads to the development of the Open Source Quality Radar (OSQR) [22] for facilitation of the DevOps team. By design OSQR was developed as a service which can be shared within the Group IT. The FOSS component focused approach addresses that optimized algorithms in libraries and frameworks are mostly by design more efficient than a self-developed algorithm – think, e.g., about compression, cryptography which is a non-trivial domain and should be handled in a professional and efficient way. Furthermore, it reduces the allocation of engineering resources for the topic by reusing existing and proven in use components.

5 Conclusion and Outlook

This work shows how enterprise IT agile value stream teams can evolve their sustainability engineering capabilities. It is important that the teams have the freedom to develop their sustainability skills. This has to be ensured by the organizations culture and habits to foster team autonomy and degrees of freedom. The examples show that sustainable software engineering can be developed and shared by doing respective operational delivery. An insight is that not always an explicit and expensive project for method development is needed. Important is that the team culture includes a higher agile mindset with established collaboration and sharing principles. A limitation within the shift-left approach is a separation of dev and ops. Because, as the example shows, the shift-left as the lever for high impact of small actions was only possible within the DevOps team.

The key contributions *to practice* can be summarized by the following aspects:

- With DevOps a team setup to establish a holistic sustainability approach for their value stream respective product or service is given
- Organizations can initialize sustainability software engineering by offering adequate degrees of freedom to the DevOps team
- The main effects in sustainability come with a shift-left from run to build.
- In non-DevOps organizations an initial initiation in the ops-team can be limited by the silo border to the dev-team.

The key contributions *to theory* can be summarized by the following aspects:

- Identification that sustainability can become a quality characteristic of software
- Identification that a holistic shift-left approach is needed for sustainability impact
- Identification that the organizational refinement aspects and the software life-cycle have to be thought together.
- Identification that different agile organizational structures like dev-teams or DevOps-teams require different approaches for effective sustainability engineering

As summary about sustainable IT artifacts should be a shared responsibility between IT and workload owner: *IT responsibility is sustainable service delivery* and *user responsibility is sustainable service consumption*. This includes that the IT cares about a sustainable software development and operation and the user respective workload owner cares about a sustainable usage of the IT. A precondition respective assumptions is that the IT artifact itself is a valuable artifact within an overall sustainable business context.

Also keep in mind that currently often *sustainability is measured outside* – e.g. by an (external) audit - in the operating phase but mostly *sustainability is decided inside* IT during design and implementation in an earlier development phase. A potential measured derivations to the expectations cannot be “healed” at or after the point of the late measurement without costly and resource intensive refactoring or re-implementation of the software. Therefore, *software sustainability is a build-in quality characteristic*.

An investigation aspect for the future is how to facilitate holistic improvements of sustainability with separated dev and ops teams by establishing collaborative goals over the different teams. The Volkswagen Group IT still starts different new initiatives which foster green IT and sustainability. Based on these triggers further building blocks for sustainable software engineering will be developed in the near future. However, there is still a lot of work to do to develop skills and capabilities for sustainable software engineering with the proposed shift-left mindset in all the value stream teams.

References

1. Lozano, R.: Towards better embedding sustainability into companies’ systems: an analysis of voluntary corporate initiatives. *J. Clean. Prod.* **25**, 14–26 (2012)
2. Jose, P.D., Saraf, S.: Corporate sustainability initiatives reporting: a study of India’s most valuable companies. *Corporations Sustain.*, 49–88. Routledge (2017)
3. Ismail, N.B., Alcouffe, S., Galy, N., Ceulemans, K.: The impact of international sustainability initiatives on Life cycle assessment voluntary disclosures: the case of France’s CAC40 listed companies. *J. Clean. Prod.* **282**, 124456 (2021)
4. ISO 14001:2015 Environmental management systems — Requirements with guidance for use
5. Naumann, S., Dick, M., Kern, E., Johann, T.: The greensoft model: a reference model for green and sustainable software and its engineering. *Sustain. Comput.: Inform. Syst.* **1**(4), 294–304 (2011)
6. Barroero, T., Motta, G., Durante, M.: Sustainable service level agreements. *IEEE SCC*, 679–684 (2011)
7. Macías, M., Guitart, J.: SLA negotiation and enforcement policies for revenue maximization and client classification in cloud providers. *Future Gener. Comput. Syst.* **41**, 19–31 (2014)
8. Sengupta, J., Singh, P., Suri, P.K.: Energy aware next fit allocation approach for placement of VMs in cloud computing environment. In: Arai, K., Kapoor, S., Bhatia, R. (eds.) *FICC 2020*. AISC, vol. 1130, pp. 436–453. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-39442-4_33
9. MacDonald, C.: Understanding participatory action research: a qualitative research methodology option. *Can. J. Action Res.* **13**(2), 34–50 (2012)
10. UN SDG: <https://sdgs.un.org/goals>. Access validated 18 Jul 2022
11. Behrendt, S., Göll, E., Korte, F.: Effizienz, Konsistenz, Suffizienz: strategieranalytische Betrachtung für eine Green Economy. Institut für Zukunftsstudien und technologiebewertung,

- IZT-Text 1–2018, ISBN: 978–3–941374–35–5 (2018). https://www.izt.de/fileadmin/publikationen/IZT_Text_1-2018_EKS.pdf. Access validated 08 Jun 2022
12. Belady, C., Rawson, A., Pfeuger, J., Cader, T.: Green grid data center power efficiency metrics: PUE and DCIE. The green grid, 1–9 (2008)
 13. Koomey, J., Berard, S., Sanchez, M., Wong, H.: Implications of historical trends in the electrical efficiency of computing. *IEEE Ann. Hist. Comput.* **33**(3), 46–54 (2010)
 14. Plan-Build-Run approach: <https://www.mckinsey.com/business-functions/mckinsey-digital/our-insights/using-a-plan-build-run-organizational-model-to-drive-it-infrastructure-objects>. Access validated 18 Jul 2022
 15. Poth, A., Nunweiler, E.: Develop sustainable software with a lean ISO 14001 setup facilitated by the efiS@ Framework. In: Przybyłek, A., Jarzębowicz, A., Luković, I., Ng, Y.Y. (eds.) *LASD 2022. LNBIP*, vol. 438, pp. 96–115. Springer, Cham (2022). https://doi.org/10.1007/978-3-030-94238-0_6
 16. 1-hour project: <https://www.volkswagenag.com/en/sustainability/strategy-policy-engagement/engagement/project-one-hour.html>. Access validated 08 04 2022
 17. Go2zero: <https://www.volkswagenag.com/en/news/stories/2019/07/co2-getting-to-zero.html>. Access validated 18 Jul 2022
 18. Poth, A., Schubert, N., Riel, A.: Sustainability efficiency challenges of modern it architectures – a quality model for serverless energy footprint. In: Yilmaz, M., Niemann, J., Clarke, P., Messnarz, R. (eds.) *Systems, Software and Services Process Improvement: 27th European Conference, EuroSPI 2020, Düsseldorf, Germany, September 9–11, 2020, Proceedings*, pp. 289–301. Springer International Publishing, Cham (2020). https://doi.org/10.1007/978-3-030-56441-4_21
 19. Poth, A., Kottke, M., Heimann, C., Riel, A.: The EFIS framework for leveraging agile organizations within large enterprises. In: Gregory, P., Kruchten, P. (eds.) *XP 2021. LNBIP*, vol. 426, pp. 42–51. Springer, Cham (2021). https://doi.org/10.1007/978-3-030-88583-0_5
 20. Poth, A., Kottke, M., Middelhave, K., Mahr, T., Riel, A.: Lean integration of IT security and data privacy governance aspects into product development in agile organizations. *J. Univ. Comput. Sci. (JUCS)* **27**(8), 868–893 (2021)
 21. Poth, A., Widock, A., Henschel, A., Eissfeldt, D.: Foster sustainable software engineering awareness in large enterprises – from a cheat-sheet for technical and organizational indicators to dashboards, euroSPI'22, Springer, in publication process (2022) https://doi.org/10.1007/978-3-031-15559-8_5
 22. Poth, A., Levin, D-A., Rrjolti, O., Wanjetscheck, M.: Quality evaluation with the open-source quality-radar for a sustainable selection and use of FOSS components, euroSPI'22, Springer, in publication process (2022). https://doi.org/10.1007/978-3-031-15559-8_36

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.



3rd International Workshop on Agility and Microservices



Improving the Implementation of Microservice-Based Systems with Static Code Analysis

Sebastian Copei^(✉), Maximilian Schreiter, and Albert Zündorf

Kassel University, Kassel, Germany

{sco, zuendorf}@uni-kassel.de, maximilian.schreiter@t-online.de

Abstract. IDE's (Integrated Development Environments) are powerful tools to support the development process and help to write better and cleaner code. However, most IDEs aim with their support on the development of monolithic software. For the development of distributed systems like a microservices-based system, there is a lack of IDE support for e.g. inter-service communication via REST. This means through the string-based communication used during REST, the IDE is not able to support with autocompletion, syntax highlighting or type checking. To solve this issue, we introduce an IDE plugin called **SIARest (Software Interface Analyser)**, based on the LSP (Language Server Protocol), which gives additional support during the development of distributed microservice-based systems, in a **monorepo** environment.

Keywords: microservices · static analysis · agile development

1 Introduction

In this paper, we will introduce a new IDE plugin hereinafter referred to as **SIARest**, which enables commonly used IDE features, like type checking, syntax highlighting and autocompletion, to the microservice cosmos. Currently, these features are limited to monolithic applications. In this chapter we will describe our motivation and highlight the goals of this paper through research questions, also we will give a short overview of the related work. In the second chapter, we will introduce some basic knowledge. Afterwards, in the third chapter, **SIARest** will be presented. The fourth chapter will summarize the presented knowledge and answers the research questions of this paper. The final chapter will give a short outlook to the upcoming work.

1.1 Motivation

Writing code documentation is often seen as chores. But, without a proper documentation, the usage of a REST API is nearly impossible. The problem that

comes with tools like **Swagger** (see Sect. 1.3) is that the generated documentation needs to be synchronized with the written code manually. Even if **Swagger** is able to generate an API service from an **OpenAPI** description, this code needs to be adjusted manually. This in turn brings deviation between code and documentation. From the idea to automatically check the API documentation against the written code, the way was short to come up with an IDE integration through a plugin which can check an implemented API service against an API specification.

In Sect. 1.2, we discuss other solutions and contrast them with our own. The Sect. 2 will introduce basic knowledge for the further descriptions. After this, in Sect. 3, we briefly introduce the actual implemented plugin. The conclusion and the future work are presented in Sect. 4 and 5.

1.2 Related Work

The most tooling for development of REST services are interactive testing tools like **Postman**¹ or **Curl**². They are used after a REST API is implemented to check the correctness of the API through predefined testing scenarios. In comparison to our plugin, these tools are not directly integrated into the development cycle, because they are not directly integrated into the IDE to give direct feedback about the implementation. However, **Visual Studio Code** offers a plugin³ for API testing from the IDE. But, the problem of the lack of integration into the development cycle is still present.

Another tool which is often used for REST endpoints is **Swagger**⁴. With **Swagger**, it is possible to describe a REST API with the **OpenAPI**⁵ specification. From this specification, **Swagger** can generate an online documentation for a REST API. Even more, it is possible to try the API in a generated web page, which could be served through any web server. **Swagger** can be integrated into the development cycle through annotations. This is available for most backend libraries in various languages like **Spring Boot** (Java) [13] or **NestJS** (TypeScript) [9]. Through this, it is possible to generate the **Swagger** documentation and web page directly from the code. However, this process has no IDE support. The IDE will not highlight deviated descriptions from an annotation with the actual implemented code. **SIARest** is able to highlight wrongly implemented endpoints.

In ‘‘**SafeRESTScript: Statically Checking REST API Consumers**’’, Nuno Burnay et al. presents a new language which should allow static validation of REST endpoints [3]. The language is called **SafeRESTScript** and leans syntactically on **JavaScript**. The specification for validation is written in **HeadREST** [14], which can be compared to the **OpenAPI** specification used by **Swagger**. The approach of Nuno Burnay et al. relies on model checking methods. They created a new language to

¹ <https://www.postman.com/>.

² <https://curl.se/>.

³ <https://marketplace.visualstudio.com/items?itemName=humao.rest-client>.

⁴ <https://swagger.io/>.

⁵ <https://www.openapis.org/>.

write REST endpoints. For their ecosystem, this may be a good strategy to ensure correctness of REST API. However, this also means that developers need to reimplement their services with `SafeRESTScript` and also need to specify their services with `HeadREST`. This is a very hard change of the development circle. `SIARest` can be used after the implementation to check for correctness, a reimplementation is not needed.

Another formal approach is presented in `Formal Verification of Stateful Services with REST APIs Using Event-B` by Irum Rauf et al. [11]. The authors use the `Event-B` [1] modeling tool to address inconsistency design issues, model checking of service specifications and the state-explosion problem [11]. Their solution could be integrated into a continuous integration pipeline. In contrast to our plugin, the pipeline does not provide feedback instantaneous to the developer.

Beside the formal approaches, there are more practicable tools which provide a similar set of features than our plugin. In the paper `Statically Checking Web API Requests in JavaScript`, the authors also created a tool which is able to check an API specification against an implementation with `JavaScript`, `JQuery`⁶ and `Ajax`⁷ [15]. The same authors have integrated this tool into the IDE `Atom` [16]. Regardless, this integration is a popup which shows current problems with the actual API implementation. The plugin does not provide direct interaction with the developer through well known IDE features like syntax highlighting.

2 Basics

This chapter will give a brief introduction to the `Monorepo` pattern and the `Language Server Protocol`. We will describe shortly the technologies and the usage in the context of `SIARest`.

2.1 Monorepo

The `Monorepo` is a pattern for project organization. Rather than managing every project in its own `Git` repository, a `Monorepo` controls all projects in a single `Git` repository at once [4]. Although, the code is managed by one repository, each service still can be developed and deployed independently [6].

Beside the technical advantages of `Monorepos` like one source of truth, code reuse, atomic changes [7], there are social benefits. The developers work together on a single product in a single repository without any restriction of visibility or access control. This can improve the cohesion of a team [2]. However, the downsides of `Monorepos` are lack of access control, long build times and the bad performance of `git` on huge repositories [7]. The fact that big companies like `Google` [10], `Facebook` [8] and `Microsoft` [5] also use `Monorepos` shows that this pattern is production ready.

⁶ <https://jquery.com/>.

⁷ <https://api.jquery.com/jquery.ajax/>.

To use the full feature set of `SIARest`, we currently recommend organizing the code of multiple services in a `Monorepo`. This regards to the feature of showing of and jumping to REST API methods.

2.2 Language Server Protocol

The `Language Server Protocol`⁸ defines the communication between an editor or IDE and a `Language Server`. It was created and maintained by Microsoft. The source code is available on GitHub in the Microsoft organization⁹.

The `Language Server` runs as a separate child process of an editor or IDE. The communication between parent and child process takes place using the `JSON-RPC` protocol and works similar to REST APIs [12]. Therefore, it standardizes what is received by the `Language Server` and what is sent back to the editor or IDE.

Many coding features like autocompletion, `GoTo` definition, or documentation on hover for a programming language are supported by the protocol. It is used by multiple different `Language Servers` like the `CSS Language Server`¹⁰ or the `TypeScript Language Server`¹¹ which are included in `Visual Studio Code`.

3 SIARest

Currently, `SIARest` is only working for `Express` backends and `Angular` frontends written in `TypeScript`.

`SIARest` is an extension for `Visual Studio Code`¹². It implements the `Language Server Protocol` to provide coding features for the development of web applications based on `TypeScript` and `Express`¹³. These features includes syntax highlighting, autocompletion, `GoTo` definitions, and hover infos.

`SIARest` consist of three parts. The extension, which is necessary to communicate with `Visual Studio Code` and the `Language Server`. The `Language Server`, resolves the request redirected from `Visual Studio Code` by the extension. A configuration file, which contains definitions of endpoints for the corresponding web application.

For simplicity, we refer `Visual Studio Code` and the extension as the `client` and the `Language Server` as the `server`.

The `client` and the `server` use the `Language Server Protocol` to exchange requests and responses. Requests like opening, editing or saving a file

⁸ <https://microsoft.github.io/language-server-protocol/specifications/specification-current/>.

⁹ <https://github.com/microsoft/language-server-protocol>.

¹⁰ <https://github.com/Microsoft/vscode/tree/main/extensions/css-language-features/server>.

¹¹ <https://github.com/typescript-language-server/typescript-language-server>.

¹² <https://code.visualstudio.com/>.

¹³ <https://github.com/expressjs/express>.

trigger the analysis process for `TypeScript` files. Afterwards, the server transforms the associated file to an abstract syntax tree. For this transformation, we use the `TypeScript Compiler API`¹⁴.

Next, we parse the abstract syntax tree and search for typical phrases of the `Express` framework. These phrases could be expressions like `app = express()`;¹⁵ or `router = Router()`;¹⁶.

If such phrases are found, we can expect that this file uses the web framework and we start with a deep structural analysis. During the deep analysis, we are looking for expressions that define API endpoints. To be precise, we are looking at HTTP methods, like `GET`, `POST` or `DELETE`.

If there are endpoints present, we cache the results to be able to analyze them in more detail. Since `SIARest` provides support for syntax highlighting, we want to show errors or warnings directly after the user is editing a file. Because of this, we start to check the cached results for errors or warnings with the help of our configuration file directly after traversing the abstract syntax tree.

The following Fig. 1 shows an example for the syntax highlight which is provided by `SIARest`.



Fig. 1. Syntax Checking

To know which values are wrong, our configuration file is used. This file is a JSON file and holds information about the API and its endpoints. In a nutshell, the configuration consists of multiple service description which consist of multiple endpoint definitions. The syntax is inspired by the `OpenAPI` specification, but is simplified and minimized. Currently, this configuration file needs to be written by hand. In Chap. 5 we discuss a possible solution for this issue.

Further features like `GoTo` definitions or reference search for API URLs are also supported. At the moment, these features are only available for `monorepos`. In both features, `SIARest` searches for the usages of API URLs from an express project in `Angular` frontends and vice versa. Through this, it is possible to ‘‘jump’’ between the definition and the reference of API endpoints. To find these references and definitions the cached results of the analysis process are used.

¹⁴ <https://github.com/microsoft/TypeScript/wiki/Using-the-Compiler-API>.

¹⁵ <http://expressjs.com/en/4x/api.html>.

¹⁶ <http://expressjs.com/en/4x/api.html#router>.

The **Hover** feature shows specific information about endpoints inside the editor. It uses the information from the configuration file and creates a readable string that shows information when hovering over an API address. The information contains the URL Address, type of the endpoint, and both types of the **request** and/or **result** object¹⁷.

The last feature is the autocompletion. It takes the information from each entry of the configuration file and takes the address to suggest it for autocompletion. This is shown in Fig. 2.

The screenshot shows a code editor with a dark theme. On line 22, the code is `this.httpClient.get('rest');`. A dropdown menu is open below the code, listing four suggestions: `/rest/article` (with a `string` type), `/rest/shoppingcart`, `/rest/shoppingcart`, and `/rest/uploadData`. A mouse cursor is hovering over the second `/rest/shoppingcart` suggestion.

Fig. 2. Autocompletion

For the sake of replication, you can find **SIARest**¹⁸ and a simple example project¹⁹ with a running configuration on GitHub.

4 Conclusion

In this paper, we introduced the **Visual Studio Code** plugin **SIARest**, which brings common coding support to the microservice-based development of web applications. Unfortunately, we have not tested the plugin in a bigger manner with more than several test users. Hence, we want to bring a first version of the plugin into the **Visual Studio Code Marketplace** to reach a bigger user base and to collect feedback for further improvements.

5 Future Work

For our next steps, we want to remove the necessity of a configuration file. This could be achieved through the integration of an already existing specification like **OpenAPI**. This also ensures that the API documentation never deviate from the actual implementation, because of the type checking during development time. Furthermore, we want to erase the requirement of the **monorepo** to use all features of **SIARest**. **SIARest** should be easy as possible, so the integration barrier is as low as possible for developers. Finally, **SIAREST** needs to integrate more

¹⁷ https://github.com/siatoolsuit/BusinessTrip/blob/master/business_trip-backend/siarc.json.

¹⁸ <https://github.com/siatoolsuit/siarest-vscode>.

¹⁹ <https://github.com/siatoolsuit/BusinessTrip>.

libraries and technologies like `SpringBoot` (Java) or `NestJS` (TypeScript) and also should support more IDEs than `Visual Studio Code`. Because of the usage of the `Language Server Protocol`, the integration of `SIARest` into other IDEs should be easy.

References

1. Abrial, J.-R.: *Modeling in Event-B: System and Software Engineering*. Cambridge University Press (2010). <https://doi.org/10.1017/CBO9781139195881>
2. Brousse, N.: The issue of Monorepo and Polyrepo in large enterprises. In: *Proceedings of the Conference Companion of the 3rd International Conference on Art, Science, and Engineering of Programming, Programming 2019*, New York, NY, USA, 2019. Association for Computing Machinery. <https://doi.org/10.1145/3328433.3328435>
3. Burnay, N., Lopes, A., Vasconcelos, V.T.: Statically checking REST API consumers. In: de Boer, F., Cerone, A. (eds.) *SEFM 2020*. LNCS, vol. 12310, pp. 265–283. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-58768-0_15
4. From monolith to monorepo. <https://medium.com/@brockreece/from-monolith-to-monorepo-19d78ffe9175>. Accessed 22 Mar 2022
5. Microsoft now uses git and GVFS to develop windows. <https://techcrunch.com/2017/05/24/microsoft-now-uses-git-and-gvfs-to-develop-windows/>. Accessed 22 Mar 2022
6. Misconceptions about monorepos: Monorepo != monolith. <https://blog.nrwl.io/misconceptions-about-monorepos-monorepo-monolith-df1250d4b03c>. Accessed 22 Mar 2022
7. The pros and cons of monorepos, explained. <https://betterprogramming.pub/the-pros-and-cons-monorepos-explained-f86c998392e1>. Accessed 12 Jul 2022
8. Scaling mercurial at facebook. <https://engineering.fb.com/2014/01/07/core-data/scaling-mercurial-at-facebook/>. Accessed 22 Mar 2022
9. OpenAPI integration in NestJS. <https://docs.nestjs.com/openapi/introduction>. Accessed 22 Mar 2022
10. Potvin, R., Levenberg, J.: Why google stores billions of lines of code in a single repository. *Commun. ACM* **59**(7), 78–87 (2016). <https://doi.org/10.1145/2854146>
11. Rauf, I., Vistbakka, I., Troubitsyna, E.: Formal verification of Stateful services with rest APIs using event-B. In: *2018 IEEE International Conference on Web Services (ICWS)*, pp. 131–138 (2018). <https://doi.org/10.1109/ICWS.2018.00024>
12. Language server protocol overview. <https://microsoft.github.io/language-server-protocol/overviews/lsp/overview/>. Accessed 26 Mar 2022
13. Setting up swagger 2 with a spring REST API. <https://www.baeldung.com/swagger-2-documentation-for-spring-rest-api>. Accessed 22 Mar 2022
14. Vasconcelos, V.T., Martins, F., Lopes, A., Burnay, N.: HEADREST: a specification language for RESTful APIs. In: Boreale, M., Corradini, F., Loreti, M., Pugliese, R. (eds.) *Models, Languages, and Tools for Concurrent and Distributed Programming*. LNCS, vol. 11665, pp. 428–434. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-21485-2_23

15. Wittern, E., Ying, A.T.T., Zheng, Y., Dolby, J., Laredo, J.A.: Statically checking web API requests in JavaScript. In: 2017 IEEE/ACM 39th International Conference on Software Engineering (ICSE), pp. 244–254 (2017). <https://doi.org/10.1109/ICSE.2017.30>
16. Wittern, E., et al.: Opportunities in software engineering research for web API consumption. In: 2017 IEEE/ACM 1st International Workshop on API Usage and Evolution (WAPI), pp. 7–10 (2017). <https://doi.org/10.1109/WAPI.2017.1>

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.





Towards an Architecture-Centric Methodology for Migrating to Microservices

Jonas Fritzschi^{1(✉)}, Justus Bogner³, Markus Haug¹, Stefan Wagner¹,
and Alfred Zimmermann²

¹ University of Stuttgart, Stuttgart, Germany
{jonas.fritzschi,markus.haug,
stefan.wagner}@iste.uni-stuttgart.de

² University of Applied Sciences Reutlingen, Reutlingen, Germany
alfred.zimmermann@reutlingen-university.de

³ Vrije Universiteit Amsterdam, Amsterdam, The Netherlands
j.bogner@vu.nl

Abstract. The euphoria around microservices has decreased over the years, but the trend of modernizing legacy systems to this novel architectural style is unbroken to date. A variety of approaches have been proposed in academia and industry, aiming to structure and automate the often long-lasting and cost-intensive migration journey. However, our research shows that there is still a need for more systematic guidance. While grey literature is dominant for knowledge exchange among practitioners, academia has contributed a significant body of knowledge as well, catching up on its initial neglect. A vast number of studies on the topic yielded novel techniques, often backed by industry evaluations. However, practitioners hardly leverage these resources. In this paper, we report on our efforts to design an architecture-centric methodology for migrating to microservices. As its main contribution, a framework provides guidance for architects during the three phases of a migration. We refer to methods, techniques, and approaches based on a variety of scientific studies that have not been made available in a similarly comprehensible manner before. Through an accompanying tool to be developed, architects will be in a position to systematically plan their migration, make better informed decisions, and use the most appropriate techniques and tools to transition their systems to microservices.

Keywords: microservices · refactoring · software architecture

1 The Challenge of Moving to Microservices

In times of cloud-based software solutions, the microservices architectural style has become the de facto standard for large-scale and cloud-native commercial applications [15]. Technological advancements like containerization and automation have paved the way for efficiently operating almost any number of independent functional units. However, existing legacy systems are often designed

as monoliths and can therefore barely benefit from advantages such as improved scalability, maintainability, and agility through independent deployment units [11]. Hence, many companies try to migrate their systems towards microservices. While a rewrite of the entire application is expensive and often infeasible, architects are looking for less resource-intensive approaches to modernize a system. Architectural refactorings that are (partly) automated may reduce effort and risk of a migration tremendously. Unfortunately, there is no general approach that fits for arbitrary systems [10]. A thorough analysis is required to choose the appropriate strategy and refactoring technique.

Early adopters of microservices had to deal with manifold challenges, such as a lack of technical guidance and best practices, immature tooling, or organizational aspects. While pioneers like Amazon, Netflix, Spotify, and even the German retailer Otto published on their journeys of microservices adoption, such exemplary cases do not necessarily qualify as a blueprint. The average enterprise system has often more sophisticated tasks to fulfill than the well-studied retail domain, and moreover, IT budgets are often tight. As well, strict compliance and high-quality requirements do not leave much room for experimentation and failed investments. Hence, architects often struggle to find suitable guidance on planning and conducting such an architecture change systematically. A migration and in particular the decomposition of industry-scale legacy systems is seen as major challenge in this regard [9]. While practitioners often achieve a reasonable solution with extensive manual efforts, the targeted and quality-controlled planning of a migration as well as a semi-automated decomposition continue to be problematic. In the following two subsections, we briefly summarize our view on the typical progress of a system migration to microservices and describe the gap between academia and industry.

1.1 Three Phases of a Migration

Based on our groundwork and existing research [5, 14, 16], we identify three main phases of a migration: *system comprehension*, *planning*, and *implementation*. The initiation of a migration is commonly started with comprehending the existing system: After the definition of strategic goals, quality requirements are determined by all stakeholders of the system. They serve as a measure for assessing the legacy system and potential alternatives. Hence, the outcome of this first phase should be a grounded decision for or against a modernization, based on specific quality attributes and metrics. While monolith and microservices are not the only possible architectural styles, we exclusively focus on the contraposition of these two patterns.

Given that the comprehension phase resulted in favor of a migration, the planning phase aims at defining an adequate strategy. One of the two major tasks in this phase is the definition of a development process based on different types of software modernization [3]. Distinguishing greenfield and brownfield develop-

ments¹, we further split up the second in a re-build or re-factor development type. Further distinctions can be made that affect the time frame and consumption of resources. While a *big bang*² migration aims to minimize the duration, a continuous evolution strategy tries to minimize the needed resources. The second decision in the planning phase concerns the choice of a service identification approach. It yields a suitable service cut (decomposition of the existing system) and thereby determines the granularity of resulting services. Our previous research has shown that deciding on the decomposition is often a manual task [9] guided by methods like Domain-Driven Design (DDD). However, a variety of existing artifacts from the legacy system can be beneficial for automating this task to some extent, e.g., code bases, databases, version control system data, runtime logs and traces, and various other design documents or models.

After completion of the initial two phases, the actual implementation starts. The elaborated strategy implies boundaries for duration, required resources, as well as needed organizational changes. The latter are in particular relevant as a consequence of altering a system’s architecture [7]. Microservices candidates and target architecture are defined, based on the approach and techniques chosen in the planning phase. The now following implementation of services commonly iterates through several cycles. In each cycle, one or more microservices are implemented, accompanied by a quality assessment of the emerging target system. That way, inadequacies of the architecture definition or even an unsuitable decomposition can be corrected early with reasonable effort. The organizational changes, infrastructure build-up, and establishment of DevOps processes go hand in hand with the migration progress.

1.2 The Academia-Industry Gap

A rapidly growing number of scientific publications deal with the topic of microservices migrations, as the meta studies by Schroer et al. [13] and Ponce et al. [12] show. Existing research covers a variety of topics, starting from decision-making over process strategies [3] to quality assurance [6] and organizational aspects [9]. The challenging question of service identification techniques in general [1] and for microservices specifically is targeted by several dozen studies [10,12]. However, our empirical research has shown that this extensive body of scientific literature is mostly unknown to practitioners and therefore rarely leveraged [9]. We found that even specialized consultancy companies do rarely consider such knowledge. There may be several aspects to this barrier, e.g., reservations regarding scientific databases, access limitations, or concerns regarding the practical applicability and relevance of scientific research. Hence, a key motivation of our work is in filtering, pre-processing, and presenting the relevant works to practitioners based on their specific systems and migration scenarios.

¹ Greenfield development refers to creating a system for a new environment from a clean slate, no legacy code is required. Brownfield developments require the presence of an existing system that gets improved: data, processes and settings are retained.

² Migration within a limited time window and instant switch from old to new system.

2 Research Design

Figure 1 illustrates the overall research method. Our research objective is framed by the following two questions:

- RQ1: How can a process framework represent a holistic view on microservice migration activities with a focus on architectural refactoring techniques?
 RQ2: How can tool support based on such a framework provide guidance for architects in a specific migration scenario?

As a foundation, we analyzed existing literature on the microservice migration process. In an interview study among 16 practitioners from 10 companies, we analyzed 14 systems from various domains regarding intentions, practices, and challenges [9]. In addition, we contributed an early meta study classifying architectural refactoring approaches for migrations to microservices [10]. Our resulting methodology serves as a basis for longitudinal case studies that are currently conducted in cooperation with industry (including DATEV eG and Siemens AG). In an iterative process, the framework and accompanying tool support will be evaluated and refined. As a final step, we plan a large-scale survey among practitioners to assess the accompanying tools' applicability in certain contexts, its usefulness, and usability.

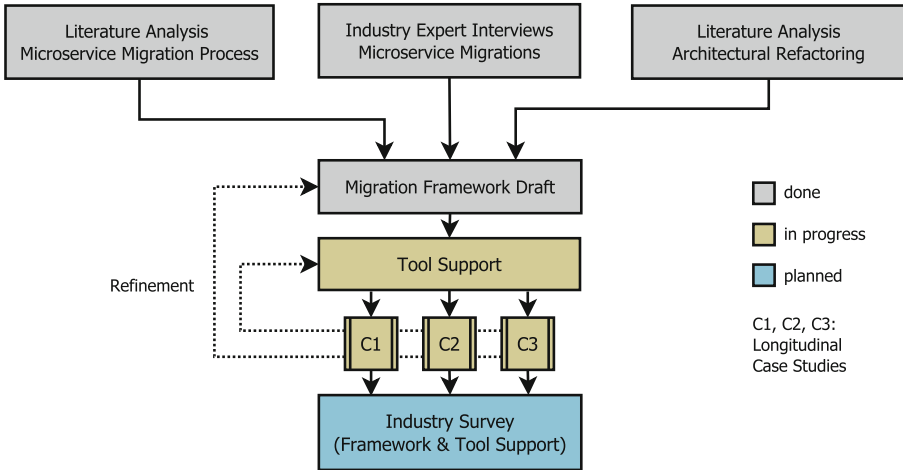


Fig. 1. Research Method

3 Related Work

According to the outlined research method, we split up the discussion of related studies into three clusters: 1) migration process, 2) architectural refactoring, and 3) associated aspects like quality assurance and re-organization.

1) Migration Process. In their survey among 18 practitioners, Di Francesco et al. collected the various activities carried out in a migration to microservices [8]. The work provides an empirically collected, bottom-up classification of common activities. Taibi et al. followed a similar survey-based approach when querying 21 practitioners [14]. They reconstructed a migration process framework that reflects the interviewees' procedures and best practices. In addition to Di Francesco et al., they also distinguish between re-development and continuous evolution strategies, applying the popular Strangler pattern. Wolfart et al. approach the migration topic more holistically in their migration roadmap [16]. They analyzed six primary studies to come up with a unified process. To this end, they conducted a more comprehensive systematic mapping of 62 primary studies dealing with the modernization of legacy systems to microservices. Their resulting framework depicts eight activities grouped into four phases, namely Initiation, Planning, Execution, and Monitoring. In the same way, we can regard the roadmap by Bozan et al. [5] on incrementally transitioning to a microservices architectures, which was distilled from interviews with 31 software experts. In contrast to the above-mentioned studies, the authors here also reflect on the organizational and business-related impacts.

2) Architectural Refactoring. The secondary studies by Abdellatif et al. [1], Bajaj et al. [3], Schroer et al. [13], Ponce et al. [12], and Fritzscht et al. [10] provide a holistic overview of this major technical challenge in a migration. Our earlier study [10] attempted a classification of approaches based on their underlying techniques. Abdellatif et al. [1] developed a more elaborate taxonomy that provides a solid foundation for use in our framework. The majority of studies can be ascribed to at least one of the three basic categories of techniques: model-driven, static analysis, and dynamic analysis [12]. In addition, organizational structures or metadata like version control history [10] can also provide valuable input for the architectural refactoring.

3) Associated Aspects. As initially set strategic goals and subsequently identified quality attributes largely steer the architecture transformation, quality assurance needs a strong focus, as outlined by Shahin et al. [13]. This aspect is reflected in some of the above suggested frameworks during the initial phase (requirements and strategic goals) or in the form of verification and validation activities. As a basis for assessing the relevance of different quality attributes for microservices in general [4] and in the context of a migration [9], we build upon our earlier empirical research. It also revealed that organizational changes and social aspects such as a mindset change can have considerable impact on the migration process.

Significant advances have been made in detailing the process of a migration, as well as in elaborating techniques for decomposing monolithic systems. However, there is no holistic methodology available that combines both aspects. In addition, our research revealed the lack of a vehicle for knowledge transfer into practice. We aim to address this issue by suggesting a methodology that presents a holistic view on microservice migration activities, with a focus on architectural refactoring. It is enriched with a systematic quality assessment and aims for a

high degree of automation. Furthermore, we seek to develop a supplemental tool that guides architects, thereby allowing them to efficiently leverage the comprehensive body of scientific knowledge.

4 Proposed Migration Framework

To address RQ1, we propose an architecture-centric framework for migrating to microservices as depicted in Fig. 2. It incorporates ideas and groundwork of existing research, especially the works by Wolfart et al. [16]. Aspects of the works by Taibi et al. [14] and Bozan et al. [5] have influenced the design as well. According to the discussion in Sect. 1.1, we split a migration into three phases that are detailed below. Each activities' result artifacts are shown in a flowchart on the right side in Fig. 2. The reflected process may be applied separately for single subsystems as required.

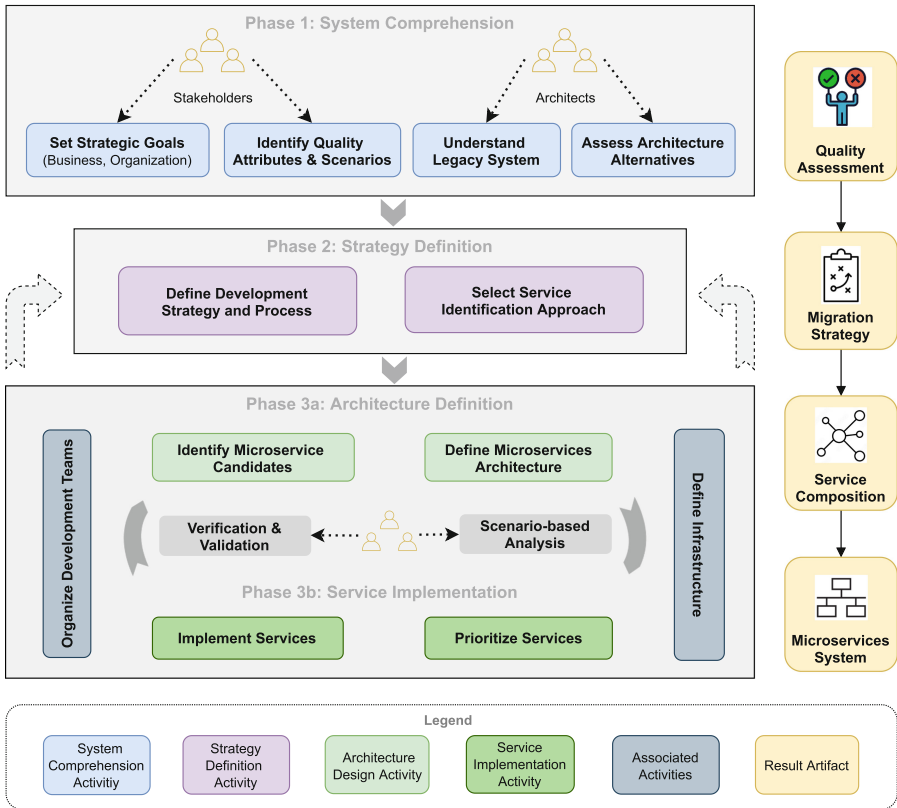


Fig. 2. Proposed Framework for Microservices Migrations

Phase 1: System Comprehension starts with a set of activities aiming to comprehend the existing system and assess alternatives as described by Wolfart et al. [16]. We depicted the common activities and involved personas. The activities in this phase are commonly performed as part of an architecture review using methods like ATAM, SAAM, or a more lightweight method, e.g., the one suggested by Auer et al. [2]. The resulting quality assessment links the decision for or against a migration to microservices to distinct scenarios and associated quality requirements which the architectural styles in question are favorable for.

Phase 2: Strategy Definition entails the two activities described in Sect. 1.1 to define the migration strategy. Depending on the system’s technological state, organizational aspects or other boundary conditions, different strategies may be chosen. The selection of a suitable approach and technique for service identification depends on several factors like targeted quality attributes, the available input artifacts, automation potential and maturity of available tool support. To this end, academia offers a variety of approaches that partly offer freely available tools that can be leveraged by practitioners.

Phase 3: Architecture Definition starts with the identification of services and a preliminary definition of the target architecture. The incremental implementation of the identified services is preceded by a prioritization step. The framework puts a major focus on quality assurance aspects to ensure that initially defined measures are applied and satisfied. Hence, the implementation activities are accompanied by a scenario-based analysis and followed by a verification & validation step. Deviations from the targets will consequently lead to altering the defined target architecture or even considering an alternative service identification approach by stepping back into phase 2.

5 Current Status of Tool Support

To address RQ2, we are in the process to develop a web-based tool that guides architects through a migration scenario. In the comprehension phase, the tool collects system specifications and guides through an architecture assessment. Based on an extensible repository of approaches for service identification and architectural refactoring, the tool will further assist in finding the appropriate technique for a specific system. As such, the repository provides selected approaches proposed by academia. In the implementation phase, the guidance will be realized in form of suggesting patterns and best practices associated with the targeted quality aspects for the migration. Methodology and tool support are currently being refined and evaluated within longitudinal industry case studies. For the framework’s structure and automation capabilities, we conducted interviews among 9 software professionals. We also see potential for hosting the developed tool publicly and expanding it by functionality to incorporate user feedback or a rating system. In that regard, the framework and tool support could facilitate knowledge transfer not just from academia to industry but also vice versa, thereby contributing to close the gap highlighted in Sect. 1.2.

References

1. Abdellatif, M., et al.: A taxonomy of service identification approaches for legacy software systems modernization. *J. Syst. Softw.* **173** (2021)
2. Auer, F., Lenarduzzi, V., Felderer, M., Taibi, D.: From monolithic systems to microservices: an assessment framework. *Inf. Softw. Technol.* **137** (2021)
3. Bajaj, D., Bharti, U., Goel, A., Gupta, S.C.: A prescriptive model for migration to microservices based on SDLC artifacts. *J. Web Eng.* (2021)
4. Bogner, J., Fritzscht, J., Wagner, S., Zimmermann, A.: Microservices in industry: insights into technologies, characteristics, and software quality. In: 2019 IEEE International Conference on Software Architecture Companion (ICSA-C), pp. 187–195. IEEE (2019)
5. Bozan, K., Lyytinen, K., Rose, G.M.: How to transition incrementally to microservice architecture. *Commun. ACM* **64**(1), 79–85 (2021)
6. Cojocaru, M.D., Oprea, A., Uta, A.: Attributes assessing the quality of microservices automatically decomposed from monolithic applications. In: 18th International Symposium on Parallel and Distributed Computing. ISPDC 2019, no. 1, pp. 84–93 (2019)
7. Conway, M.: Conway’s law (2018). <https://melconway.com/Home/Conways.Law.html>. Accessed 11 July 2022
8. Di Francesco, P., Lago, P., Malavolta, I.: Migrating towards microservice architectures: an industrial survey. In: Proceedings - 2018 IEEE 15th International Conference on Software Architecture. ICSA 2018, pp. 29–38 (2018)
9. Fritzscht, J., Bogner, J., Wagner, S., Zimmermann, A.: Microservices migration in industry: intentions, strategies, and challenges. In: IEEE International Conference on Software Maintenance and Evolution (ICSME), pp. 481–490. IEEE (2019)
10. Fritzscht, J., Bogner, J., Zimmermann, A., Wagner, S.: From monolith to microservices: a classification of refactoring approaches. In: Bruel, J.-M., Mazzara, M., Meyer, B. (eds.) DEVOPS 2018. LNCS, vol. 11350, pp. 128–141. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-06019-0_10
11. Jamshidi, P., Pahl, C., Mendonca, N.C., Lewis, J., Tilkov, S.: Microservices: the journey so far and challenges ahead. *IEEE Softw.* **35**(3), 24–35 (2018)
12. Ponce, F., Márquez, G., Astudillo, H.: Migrating from monolithic architecture to microservices: a rapid review. In: Proceedings of 38th International Conference of the Chilean Computer Science Society (SCCC 2019), Chile (2019)
13. Schröer, C., Kruse, F., Marx Gómez, J.: A qualitative literature review on microservices identification approaches. In: Communications in Computer and Information Science, vol. 1310, pp. 151–168 (2020)
14. Taibi, D., Lenarduzzi, V., Pahl, C.: Processes, motivations, and issues for migrating to microservices architectures: an empirical investigation. *IEEE Cloud Comput.* **4**(5), 22–32 (2017)
15. Vale, G., Correia, F.F., Guerra, E.M., de Oliveira Rosa, T., Fritzscht, J., Bogner, J.: Designing microservice systems using patterns: an empirical study on quality trade-offs. In: 2022 IEEE 19th International Conference on Software Architecture (ICSA), pp. 69–79. IEEE (2022)
16. Wolfart, D., et al.: Modernizing legacy systems with microservices: a roadmap. In: Evaluation and Assessment in Software Engineering, pp. 149–159. ACM, New York, NY, USA (2021)

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 in Education Tack



Being Agile in a Data Science Project

Renato Cordeiro¹(✉), Isaque Alves¹, Samara Alves², and Alfredo Goldman¹

¹ University of São Paulo, São Paulo, SP, Brazil
{renatocf, isaque.alves, gold}@ime.usp.br

² Fundação Oswaldo Cruz, Rio de Janeiro, Brazil
samara.alves@fiocruz.br

Abstract. Applying agile practices in data science requires adaptations. This paper describes challenges and lessons learned in two applied machine learning projects developed in the XP Lab course at University of São Paulo in Brazil. It compiles six suggestions for educators and practitioners who want to bring agility to their data science initiatives.

Keywords: Agile · Data Science · Machine Learning · Software Engineering · Extreme Programming · Scrum · Kanban · XP Lab

1 Introduction

Since 2001, the Institute of Mathematics and Statistics of the University of São Paulo has been offering the eXtreme Programming Laboratory (XP Lab) course. The goal of the course is to teach Agile Methods in practice [3]. Students are divided into teams and build a semester-long project for real customers.

During twelve weeks of practical activities, teams are instructed to follow the original XP practices [6]. Most teams also adopt management practices from Scrum and Kanban, learned by many students in the industry.

Given its structure, the XP Lab course provides an environment for testing the use of Agile practices in non-traditional contexts, such as in the development of the Linux kernel [2]. Since 2020, to follow the industry trend, the course organizers seek proposals for data science projects as alternatives to be developed during the course. This paper describes these experiences.

Section 2 and 3 describe the challenges and lessons learned with data science projects in the 2020 and 2021 editions of the XP Lab course. Section 4 highlights suggestions for educators and practitioners based on these experiences. Finally, Sect. 5 summarizes the main contributions from this experience report.

2 First Attempt: The Civil Police Project

In 2020, the XP Lab course organizers made their first attempt to bring data science projects to participate in the course. One proposal came from the technicians of the Intelligence Department of the São Paulo Civil Police. The goal

was to create a new tool to recognize license plates of vehicles near crime scenes, so the police could track people involved for investigations.

Given a photo captured by a security camera, students should use Machine Learning, specifically Computer Vision techniques, to separate the license plate and recognize its characters (numbers and letters). This task was particularly challenging for two reasons: photos taken by security cameras usually have low resolution and can show cars in different environments, angles, and light.

In total, six students composed the Civil Police project team. The project was successful insofar as the team delivered a demo API and built a training pipeline for a model that could receive a photo with a vehicle and output the characters from the license plate. For that, they relied on open-source libraries such as OpenCV¹ to make image transformations and TensorFlow² to train a neural network model to recognize characters.

Unfortunately, there were many challenges throughout the development. First, the team did not get access to images from the Civil Police department. Consequently, they spent lots of time collecting a dataset of photos from the internet that could emulate – albeit imperfectly – what the Civil Police technicians would collect. This affected their ability to create a model to solve the client’s actual problem.

Second, both the team did not have practical experience working with data science projects, while the Civil Police technicians did not know about Computer Vision models. As a consequence, the team spent much more time researching techniques and exploring the basics, hindering their ability to improve the model.

The first attempt with a data science project in the XP Lab course taught two important lessons. First, students should have an initial dataset to work with, or otherwise the project will be dedicated to collecting data rather than using it. Second, students should have technical guidance to help them to explore machine learning techniques and apply the data science workflow.

3 Second Attempt: The Fiocruz Project

In 2021, the XP Lab course organizers seek once again data science projects. One proposal came from the researchers of the Cellular Communication Laboratory of the Oswaldo Cruz Institute in Rio de Janeiro. The goal of the project was to create a new tool to complement the Fiocruz researchers’ ongoing effort to identify emerging technologies in scientific papers.

Given a set of articles, the Fiocruz project team should use Machine Learning, specifically Natural Language Processing techniques, to identify tech-related terms from a set of preselected articles. For that, the new tool has to cluster words from documents. Therefore, the problem requires using unsupervised learning algorithms, such as topic models, to be solved. Since the researchers were familiar with this set of techniques, they could help the team during their development.

¹ <https://opencv.org>.

² <https://tensorflow.org>.

Based on the previous experience, the XP Lab course organizers guided the Fiocruz researchers to do preparations before the project started. Particularly, the researchers built a web crawler to compile a dataset so the team could start working on the project without concerns about data collection.

In total, 17 out of 48 students were interested in the project. After the selection, six students composed the Fiocruz project team. The team reported that they felt there was a lot of value and purpose in uniting technology with the health area, learning about how they could use data science to help with this research field. They also noted that having no previous experience with Machine Learning was a contributing factor in their choice.

3.1 Development Process

The Fiocruz team developed its data science project experimentally and incrementally, following the steps described by CRISP-DM [4]. The report below describes the main activities made by the team during each development sprint, up to the end of the course. In total, there were eight one or two-week-long sprints, in which the team worked on average eight hours a week.

Sprint 1 focused on understanding the problem proposed by Fiocruz researchers and the data they provided. This sprint was used as a preparation for the team, so there was no software deliverable for the clients. The main goal was to identify the requirements and analyze what the data could offer. First, the team split into pairs to analyze the data, with multiple people doing the same task. Then, the team did Mob Programming to discuss insights, identify inconsistencies, and report discoveries about the data. In the end, the team prototyped their first data processing functions.

After collecting feedback from Fiocruz researchers, Sprint 2 focused on consolidating the data processing. The team reimplemented their prototype – a data pipeline – into a Python script, creating new functions based on insights gained from constant experimentation with the data. The team then started another research cycle, creating tasks to define the most viable techniques to handle the textual data. Finally, the team took the results to the Fiocruz researchers, so they could assist them in choosing the best tools for the job.

With a data processing pipeline mature enough, Sprints 3 and 4 consisted of exploring and applying the techniques and tools discussed previously. The Fiocruz team improved their text preprocessing using spaCy³. After that, they carried out experiments that resulted in implementing the TF-IDF (Term Frequency, Inverse Document Frequency) algorithm, a statistic that reflects how important a word is to a document in a collection of terms.

The team started an exploratory analysis of outliers based on the number of tokens, allowing them to further clean the provided dataset. It resulted in new parameterized functions to remove outliers. In parallel, the team defined activities to study the application of unit tests in the project's context, promoting new discussions within the group.

³ <https://spacy.io>.

Sprint 5 had the goal of delivering the data pipeline. The focus was to study and apply feature engineering, dimensionality reduction, and other techniques to improve the results achieved so far. Meanwhile, the team started studying Latent Dirichlet Allocation models [8]. Following the XP Lab course requirements, the team also promoted a refactoring day, which consisted of a Mob Programming session to define the project’s architecture and organize the repository.

The remaining sprints focused on applying and improving the LDA-based model, besides studying patterns to design a library that could assist Fiocruz researchers using it. After this research, the team applied the Faade design pattern [7] to create an API to access the implemented functions. Furthermore, as required by the course and in agreement with the researchers, the team created a documentation for the project, including context, architecture diagrams, and details about the architectural decisions. All artifacts can be found in the project’s repository, with an OSS license and a guide for contributions.⁴

3.2 Adapting Agile Practices

The Fiocruz project team started their development using practices from three agile methodologies: XP, Scrum, and Kanban. The items below summarize the main adaptations made in different practices to better accommodate the particularities of an applied machine learning project:

- **Data Understanding.** Being intimate with the data and knowing what it can offer is essential for creating machine learning models [1]. Therefore, the team focused its first sprint on this task and continuously reviewed its assumptions and knowledge about the data.
- **Spikes to study techniques before using them.** As the team was inexperienced with the necessary tools and techniques for the project, they created spikes to study them and then discuss solutions before coding. Only after debating and verifying the feasibility of applying different models and libraries, they started development.
- **Sprint boundaries.** As many user stories were experimental in nature, most could not be finished in the same sprint. Even after reducing their scope, it was unattainable to fit them within a single sprint. Therefore, the team gave up trying to reduce user stories and focused on collecting feedback about their progress and course correct even if tasks were unfinished. In the end, this worked well, since the results obtained met the expectations of the Fiocruz researchers.
- **Not only working software: useful data and insights.** For a data science project, discovering new tools, gathering information, and finding insights about data was just as important as developing software with quality. Therefore, the team delivered these reports to the Fiocruz researchers.
- **Mob Programming for exploration.** Following XP Lab course recommendations, the team started using Mob Programming for team building.

⁴ Documentation (PT-BR): https://gitlab.com/labxp_fiocruz/documentation.

However, they continued applying the technique weekly to share knowledge, discuss solutions, and plan activities.

- **Pair Programming for implementation.** Pair Programming was essential to share knowledge during development. On each sprint, the priority was to form pairs that had never worked together, but also help each other with their coding skills.
- **Notebooks for experimentation, scripts for production.** All experiments began on Jupyter notebooks to validate solutions and present insights to the Fiocruz researchers. After results were deemed satisfactory, the code was reimplemented with functions in Python scripts. This provided the opportunity to apply Test-Driven Development (TDD), since the team could plan their tests while prototyping in the notebooks, and then start the script reimplementaion with them.
- **Applying a different test pyramid.** Inspired by the ideas of Continuous Delivery for Machine Learning [5], the team focused on understanding different types of tests related to machine learning, particularly regarding how to test data and training pipelines.

3.3 Challenges

Throughout the project, the Fiocruz project team had to deal with many challenges related to the machine learning product development, such as:

- **Reference Architecture.** The team did not have a reference architecture to solve the problem proposed by the Fiocruz researchers. While there are well-documented architectural patterns in more traditional domains such as web development, the team had difficulties finding a proven way to implement their solution. The team architected a library using Object-Oriented software patterns [7], considering the project context and the single responsibility principle.
- **Team Insecurities.** Given the problem proposed by the Fiocruz researchers, the team was always unsure whether results were adequate. This is a characteristic of using unsupervised learning, since there was no objective way to assert the quality of proposed models. Nevertheless, the constant interaction with the researchers helped to validate the results.
- **Sprint scope.** During the first three sprints, the team tried to increase the granularity of user stories and tasks to finish them within a single sprint. However, as the tasks were experimental by nature, it was hard to predict the necessary work time. In the end, the team chose to prioritize quality. Sprints were used to maintain continuous feedback with the researchers to ensure satisfaction and reduce the risk of not delivering what was expected.

Due to the COVID-19 pandemic context, the XP Lab course was held remotely. Although this might seem a challenge, students explored how to build interpersonal relationships through team-building dynamics and slack time.

3.4 Results

At the beginning of the project, the Fiocruz team mapped tools and practices they expected to use during the development. Then, the team created a table compiling their self-assessed familiarity with those items. Figure 1a shows their knowledge at the beginning of the project. After the initial evaluation, the team defined pairings and conducted workshops to share knowledge. Figure 1b shows their knowledge by the end of the course. Fortunately, there was a significant improvement, indicating that the team learned with the experience.

Knowledge Board (Beginning)													
Name	XP	Scrum	Kanban	Docker	Git	Python	Pandas	NLTK	SpaCy	Notion	Unit tests	E2E tests	
Student #1	😊	😊	😊	😊	😊	😊	😊	😊	😊	😊	😊	😊	
Student #2	😊	😊	😊	😊	😊	😊	😊	😊	😊	😊	😊	😊	
Student #3	😊	😊	😊	😊	😊	😊	😊	😊	😊	😊	😊	😊	
Student #4	😊	😊	😊	😊	😊	😊	😊	😊	😊	😊	😊	😊	
Student #5	😊	😊	😊	😊	😊	😊	😊	😊	😊	😊	😊	😊	
Student #6	😊	😊	😊	😊	😊	😊	😊	😊	😊	😊	😊	😊	

(a) Self-assessed knowledge collected at the beginning of the project.

Knowledge Board (End)													
Name	XP	Scrum	Kanban	Docker	Git	Python	Pandas	NLTK	SpaCy	Notion	Unit tests	E2E tests	
Student #1	😊	😊	😊	😊	😊	😊	😊	😊	😊	😊	😊	😊	
Student #2	😊	😊	😊	😊	😊	😊	😊	😊	😊	😊	😊	😊	
Student #3	😊	😊	😊	😊	😊	😊	😊	😊	😊	😊	😊	😊	
Student #4	😊	😊	😊	😊	😊	😊	😊	😊	😊	😊	😊	😊	
Student #5	😊	😊	😊	😊	😊	😊	😊	😊	😊	😊	😊	😊	
Student #6	😊	😊	😊	😊	😊	😊	😊	😊	😊	😊	😊	😊	

(b) Self-assessed knowledge collected at the end of the course.

Fig. 1. Knowledge boards comparing the Fiocruz team knowledge in different methodologies, technologies, and concepts.

The weekly meetings between the team and the Fiocruz researchers allowed continuous feedback and review of results. During these meetings, the researchers focused on guiding the team’s actions towards the project goals, while giving them the freedom to experiment with different techniques and do their research. On the other hand, the team always prepared for the meetings by bringing rich insights and making technical questions about machine learning tools.

In the end, the project was delivered with a complete product that included a Python open-source library that can be integrated in the Fiocruz researchers’ routine, with documentation that will allow the project’s continuation. All code can be found in the project’s repository, with an OSS license and a guide for future contributions⁵.

4 Suggestions for Data Science Projects

Based on the experiences described in Sects. 2 and 3, here follows a set of suggestions for educators attempting to bring data science to their agile courses (or

⁵ Code (MIT License): https://gitlab.com/labxp_fiocruz/experimentation.

agility to their data science courses). These tips may also be useful for practitioners who wish to improve the agility of their own data science projects.

- **Understand the data and what it can offer.** As recommended by CRISP-DM [4], the first step in a data science project should focus on understanding the business requirements and the available data. Having a well-scoped problem and real data was paramount for the success of the Fiocruz project in comparison with the Civil Police project.
- **Use notebooks for experimentation, scripts for production.** Jupyter notebooks are a great tool for experimentation, since they promote rapid iteration during development. However, they are not ideal for production code since they complicate applying good practices such as code versioning and testing. After using them to gain insights and collect client’s feedback, code should be reimplemented in scripts using proper traditional software engineering techniques.
- **Make tests, lots of them.** Self-tested code enables refactoring and debugging. Test-driven development further improves code quality by encouraging thinking about functionality first. Data science code may go untested because the development environment does not facilitate it (see the previous item) and because it relies on external libraries. However, automated testing is a proven software engineering technique that can and should be applied in as much data science code as possible. There is emerging literature such as CD4ML [5] that provide guidance for testing different parts of applied machine learning software.
- **Use mobbing for brainstorming, use pair for coding.** Mob and Pair Programming promote joining multiple developers in a single computer to develop. Mob Programming proved itself very useful for discussing solutions and techniques, given the whole team could share their ideas. On the other hand, Pair Programming showed itself more efficient in executing coding tasks, since it allows teams to further parallelize their work.
- **Focus on quality, not deadlines.** Dividing work into sprints (as described by Scrum) did not benefit the predictability of delivery. Many tasks, experimental in nature, leaked beyond the expected sprint boundaries. Rather than viewing sprints as deadlines for the tasks, it is better to focus on the quality of results and use sprint reviews as an opportunity for continuous feedback with clients.
- **Iterate with stakeholders to collect feedback.** Customer collaboration is one of the four values of the Agile Manifesto. This interaction is even more important for data science projects, given their dependency on data. Sharing insights with clients and further understanding business requirements and data particularities allows creating better models to solve the problem proposed.

5 Conclusion

This paper showed two experiences with data science projects in the XP Lab course offered by the Institute of Mathematics and Statistics at University of

São Paulo. It summarized the challenges and lessons learned from adapting Agile practices – particularly from XP and Scrum – for data science. These adaptations were summarized in a set of suggestions to help educators and practitioners to be agile in their data science initiatives.

There are some factors that may make it difficult to reproduce the experiences described in this research, notably the positive results from the Fiocruz project described in Sect. 3. First, the Fiocruz researchers prepared a dataset for the team. Second, the researchers were technical clients that could support the team with tools and techniques. This might not be possible for all projects, as shown by the Civil Police project described in Sect. 2.

Given the successful results and the popularity of data science, the XP Lab course organizers hope to bring other data science projects to the course, and continue compiling good practices for succeeding in them.

Acknowledgments. We would like to thank the members of the Civil Police and the Fiocruz project teams, as well as thank the Intelligence Department of São Paulo Civil Police and the Cellular Communication Laboratory of the Oswaldo Cruz Institute. This research would not be possible without them.

References

1. Isaque, A., Leonardo, L., Paulo, M., Carla, R.: Product engineering for machine learning: a gray literature review. In: 2021 IEEE/ACM 43rd International Conference on Software Engineering: WAIN 2021 - 1st Workshop on AI Engineering - Software Engineering for AI
2. de Oliveira Rosa, T., Goldman, A.: Is it possible to apply agile methods to contribute to the Linux kernel?. *Agile Processes Softw. Eng. Extreme Program. Workshops* **396**, 291–297 (2020)
3. Goldman, A., Almeida Santos, V.: Continuous Improvement of an XP Laboratory Course: An 18 year History, Experience Report, In: *Agile* (2019)
4. Wirth, R., Hipp, J.: CRISP-DM: towards a standard process model for data mining. In: *Proceedings of the 4th International Conference on the Practical Applications of Knowledge Discovery and Data Mining* (2000)
5. Sato, D., Wilder, A., Windheuser, C.: Continuous Delivery for Machine Learning (2019). <https://martinfowler.com/articles/cd4ml.html>
6. Beck, K., Andres, C.: *Extreme Programming Explained: Embrace Change* (2nd Edition). Addison-Wesley Professional (2004)
7. Gamma, E., Helm, R., Johnson, R., Vlissides, J.: *Design Patterns: Elements of Reusable Object-Oriented Software*. Addison-Wesley Longman Publishing Co., Inc, (1995)
8. Blei, D.M., Andrew, Y.N.G., Jordan, M.I.: Latent Dirichlet allocation. *J. Mach. Learn. Res.* **3**, 993–1022 (2003)

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.



Panel



The Future of Work: Agile in a Hybrid World

Dennis Mancl¹  and Steven D. Fraser² 

¹ MSWX Software Experts, Bridgewater, NJ 08807, USA
dmancl@acm.org

² Innocex, Santa Clara, CA, USA
sdfrazer@acm.org

Abstract. An agile organization adapts what they are building to match their customer’s evolving needs. Agile teams also adapt to changes in their organization’s work environment. The latest change is the evolving environment of “hybrid” work – a mix of in-person and virtual staff. Team members might sometimes work together in the office, work from home, or work in other locations, and they may struggle to sustain a high level of collaboration and innovation. It isn’t just pandemic social distancing – many of us want to work from home to eliminate our commute and spend more time with family. Are there learnings and best practices that organizations can use to become and stay effective in a hybrid world? An XP 2022 panel organized by Steven Fraser (Innocex) discussed these questions in June 2022. The panel was facilitated by Hendrik Esser (Ericsson) and featured Alistair Cockburn (Heart of Agile), Sandy Mamoli (Nomad8), Nils Brede Moe (SINTEF), Jaana Nyfjord (Spotify), and Darja Smite (Blekinge Institute of Technology).

Keywords: Agile · Collaboration · In-Person World · Hybrid World · Virtual World · Societal Needs · Software Engineering · Future of Work

1 Introduction: Panel Discussion of Agility Today

This paper reports on a panel session organized by Steven Fraser (Innocex) and facilitated by Hendrik Esser (Ericsson) to discuss XP 2022’s theme “The Future of Work: Agile in a Hybrid World.” The panel was also inspired by a recent community survey by Fraser and Mancl on “The Future of Conferences” [1] in an increasingly virtual world. In preparation for the panel, panelists were asked to consider topics including:

- As pandemic restrictions are loosened, how will work environments and Agile practices evolve, given that some workers will return to a physical office while others may desire to continue their work from home?
- What issues are related to surveillance, trust, collaboration, and networking tools?
- What can we leverage from the “Peopleware” inspired writings of Fred Brooks [2], Melvin Conway [3], and Tom DeMarco and Tim Lister [4, 5]?
- What are the key learnings and emergent best practices/hybrid for hybrid work environments for Agile organizations?

In response to the pre-panel questions, Alistair Cockburn (Heart of Agile, USA) raised several concerns:

- When people do “hybrid-agile,” have they created the worst of both worlds?
- How has something fundamentally about human interaction become a technology challenge, where tools matter, and tool-usage matters more!
- What are best-practice facilitation skills for effective virtual/hybrid collaboration?
- What is the future of work? How will current company best practices evolve?

At XP 2022, in Copenhagen, the panelists shared their experiences with virtual and hybrid working environments during the pandemic, data from software developer surveys, and key ideas from previous generations of workplace evolution. Agile values and practices were important, however the panel also considered broader issues:

- Do individuals have freedom of choice regarding their own work environment?
- What are the best ways to organize work teams so they can collaborate effectively?
- How can they best approach creative problem-solving as a team?

“Adaptation” is an important part of agility. Agile teams need to adapt to changes in their organization’s work environment. The latest change is the evolving environment of “hybrid” work, a mix of in-person and virtual staff. Team members might work in the office, from home, or in other locations, but there will always be a struggle to achieve good collaboration and innovation. The motivation for virtual work isn’t just pandemic social distancing: many of us want to work virtually from home to eliminate our commute and spend more time with family.

The panel facilitator, Hendrik Esser (Ericsson, Germany), kicked off the discussion of hybrid working with an interesting anecdote, and his story is partial proof that employees are not necessarily consistent in their beliefs. In Hendrik’s office, one person who had resisted a return to the office was mandated to return two days per week. Within one week, his reaction was “I had forgotten how great it is to meet people by the coffee machine, so I think I will come more often.” Our personal beliefs about our personal “best” work environment will likely continue to evolve.

Hendrik also noted that the panel would focus more on how to work effectively in a hybrid world – rather than debating the merits of hybrid versus in-person work.

2 What is Hybrid? Individual Choice or Team Choice?

The panel abstract described “hybrid” work as an evolving environment with a mix of in-person (traditional office) and virtual (remote) work environments.

However, Darja Smite (Blekinge Institute of Technology, Sweden) saw hybrid as two sides of a coin: individual liberty versus potential team conflicts. On one side, hybrid is about flexibility, the freedom to decide where and when to work. She believed that freedom is important. However, she also acknowledged that hybrid can be problematic. Everyone desires freedom of choice regarding workplace, but Darja cautioned, “I see teams with internal conflicts, with members who have different opinions about where they want to work and how they want to collaborate.”

Sandy Mamoli (Nomad8, New Zealand) suggested the ambiguity of going hybrid. “We think we can assume that hybrid is here to stay, but we don’t know exactly what it

is going to look like.” She explained that we will need to explore new ways of working, but we must also balance our own preferences against those of the team.

Jaana Nyfjord (Spotify, Sweden) introduced Spotify’s perspective on virtual work: “Distributed first” and “work from anywhere.” Spotify encourages a non-traditional work structure. The company desires that their staff be as creative and productive as possible and enables staff to choose their workplace structure.

Darja discussed how teams might address the conflicts of work styles. One possible solution is to reorganize teams. “We ask them to imagine reshuffling the people, letting them decide how they want to work. Some teams will be fully off-site teams, some fully on-site teams, and some hybrid teams.” It’s important “to align team membership with their preferences.”

Nils Brede Moe (SINTEF, Norway) agreed with Darja, “There needs to be some compromise.” Teams often struggle when people within a single team want so many different things. Nils indicated that experienced team members can leverage an existing network of contacts for assistance, whether they are co-located or virtual. However, new team members may work more effectively face-to-face.

Sandy discussed the consequences of compromise. “I have an unpopular opinion: I don’t think it should be 100% individual choice.” Sandy explained that she has colleagues who say, “I want to live at the beach, I want to go to the office every Wednesday, and I only think about what’s fine for me.” They may not think about the bigger picture.

But Sandy believed that a top-down mandate with a single universal rule about virtual working would be wrong. “I don’t think it should be dictated by managers. I think there needs to be a conversation, with your peers at least, about what works for us... I want to make sure the whole plan works, and that factors into my decision. It’s not just ‘me, me, me.’”

3 Establishing Trust

Alistair Cockburn (Heart of Agile, USA) was skeptical about virtual work. Teams that do not share a physical workspace may find it difficult to establish trust. Alistair suggested that organizations might address this with a “budget line-item for building trust.”

This “budget” is about improving the way teams work together. Co-located teams have group activities to support communication and trust within the team: “If you are co-located in the same city, trust is [mostly] for free. [You build trust in the team] when you have a movie night, or maybe a birthday cake. You don’t get that when you’re distributed.” If distributed or hybrid teams are going to build trust, it will cost more: company management may need to budget for travel and lodging expenses and/or multiple team-building events. When a business unit has a budget line-item for trust, the cost of a distributed team becomes more visible to management.

Hendrik agreed about the value of trust: “I think the topic of trust and psychological safety is super-important.” The question is “how” to do it in a distributed environment.

4 Teamwork, Creativity, and Execution

Agile practices are designed to reinforce the importance of good communication and collaboration. Some of the practices may even improve morale in distributed teams. Nils pointed this out early in the panel discussion. “Pair programming is a very important practice. Pair programming is what has kept people going during the pandemic, because they have pair programming sessions.” Even though everyone worked virtually, pair programming provided a sense of normality and technical interaction with others. “We found that pair programming is even better for some doing it virtually, because you aren’t disturbing others.”

Darja added more information about this from a recent study [6] interview: “One person explained that his at-home experience depended on the week. He had one week alone at home [not doing any pair programming]. He wasn’t taking showers; he was sitting alone on the sofa all day. He was having a miserable work experience, he was not disciplined, and he was very unfocused. Every other week, he was paired with someone – working together for most of the day – so he would dress, he would sit by the desk, he would be motivated to be focusing on work. It was a completely different experience for the same person.”

Most of the panelists agreed that face-to-face interaction may be necessary for design discussions and creative problem-solving sessions. Some group activities are much harder to do over a computer connection.

Alistair desired that organizations considering hybrid work, consider how to keep their people happy and productive. He gave some examples of creative face-to-face problem-solving sessions – co-design with two people working together. Most people find it valuable to be involved in some group design work, and in agile development it helps to be able to sit and discuss designs with the customer. However, there needs to be a balance. When people return to the office, “I hope we start to regulate the balance between ‘I’m OK working by myself’ or ‘This is a problem-solving session.’”

Alistair also wondered if quality issues increased during the pandemic. Alistair thought that there has been a general decline in creativity and a related decline in software quality – affecting the usability of web-based user interfaces. As a frequent flyer, Alistair lamented the poor usability of airline websites. “I’m wondering whether in the pandemic, the people designing the UIs are all doing it distributed and they never get their brains together to solve the problems.”

Jaana indicated that her company (Spotify) was already proficient with virtual design work prior to the pandemic. When the pandemic hit, Spotify had existing virtual collaboration tooling and processes in place. Teams didn’t change much the way they worked.

Sandy responded that the research literature [7] indicates that “we *are* more creative if people are together.” We spark off more ideas when we are together because the framing of a face-to-face meeting is more inclusive. We have a richer interaction with others in the room. Nils noted the complex dynamics of a creative meetings: “I think that misunderstandings is the key, that you need to have disagreements to drive the innovation.” But teams who have worked together in creative brainstorming sessions *might* be able to be reasonably productive in virtual meetings: “We see that those who made that work before seem to make it work while they are distributed.”

But Sandy also defended virtual meetings. It isn't just brainstorming and ideation that steer the creative process, Sandy explained. The same research studies have found that decision making can be better in a virtual meeting. "I found really surprising that decision making was better when people were remote, because there were no distractions – and when they had to do 'dot voting' where people would vote on 'which of those creative ideas to go for,' the results were better when they were distributed."

Alistair suggested that in brainstorming and dot voting, people tend to follow their authority figure. With virtual sessions, voting can be anonymous with "better" results since people are less influenced by hierarchy.

5 Why Come to the Office?

The panelists turned to the audience for their opinions about effective work environments. Comments included:

- Mental health. So I don't go cubby-crazy at home.
- Collaboration. Getting together with the other departments.
- Work style. I work in two completely different ways at home and in the office. At home, I am very focused on tasks and productive, whereas I go to the office because I need frequent coordination with my co-workers.
- I like to work remote since I care for my son and take him to and from school. However, I still go to the office for three or four meetings a month.
- Isolation. There were times when I went to the office and almost all my team was remote, or vice versa. It was terrible and it made me feel isolated.
- Business crisis. Hybrid seems appropriate when everything works, however if your organization struggles, hybrid is an accelerator of disfunction.

Nils also reported on data from his study [8] on "return to the office." "What people want to do has been pretty stable for the last year. Everyone wants flexibility, but everyone wants an office. Younger people want to meet other young people, and they want to do that at the office, at least in our survey."

6 Why Should I Turn on My Camera?

An interesting side topic of the panel was webcam etiquette. Alistair started the discussion. He explained that he was recently invited to give a guest lecture at a local university (University of Utah), and that there were in-person and virtual attendees for his talk. He requested that any online attendee with a question should turn on their camera: "Please do me the honor of showing your face."

One person in the audience was Alistair's son, a University of Utah student, who complained, "I don't think you realize how big of an 'ask' that was. I haven't turned on my camera in two years." Alistair was amazed to hear how students were so disconnected from human contact during the pandemic.

Darja went further. She explained that she has seen the same behavior, and not just with young students. "I have spoken to people in companies who are not that young, and they have not turned on their cameras for a number of months and haven't seen

their colleagues. I'm not surprised about the younger generation. But I was surprised to see that corporate people with seniority would not turn on their cameras. Not just for someone from a university or a guest lecturer, but for their teammates."

One explanation, from both Darja and Jaana, is that there are companies who have had to work around network bandwidth problems for their online meetings. This was an issue for many companies early in the pandemic. Staff were requested to turn off cameras due to limited bandwidth. Later in the pandemic, after companies had upgraded their networks, everyone still followed old habits.

The panelists agreed that issues of "psychological safety" may motivate people to keep their cameras off. Most do not wish to be "surveilled," eight hours a day. If a meeting is not interesting, it's easier to focus on other tasks when no one can see you. Also – working from home with background commotion, most people prefer to have their cameras off and their microphones muted.

7 How to Work Effectively in the Future

Hendrik concluded the panel with one final question. "What would be your advice to companies and managers?"

Jaana suggested that we have an uncertain future, so we should focus on making micro-improvements.

Sandy believed that we should give teams more freedom and flexibility to decide how and where to work, but not necessarily individuals.

Nils advocated for more teamwork. We need to solve tasks together and work together more. It's easier when you are in the office, but it can be made to work in a hybrid environment.

Darja admitted to being excited to see the transformations of the "Workplace of the Future" and how offices will be reinvented.

Alistair gave some simple advice: Start with two days a week in the office and use that time to maximize productivity and trust-building.

8 Summary

While the panel offered some practical advice, it reached no firm conclusions. However, it is clear that the pandemic has changed how we approach Agile today. Practices that once required in-person co-location have evolved and met the challenge of virtual networking.

Virtual and hybrid work have also created new opportunities. Technology will play a role in promoting a diverse and inclusive workforce. An increase in workplace access for women, minorities, and the disabled has been described in academic studies [9] and in popular media [10, 11]. In the US, federal regulations require employers to make "reasonable workplace accommodations" for disabled workers. Because virtual work has become more established across industry during the pandemic, legal journals have suggested how workers may qualify for mandated virtual or hybrid work arrangements [12]. The case law is still in flux, and different rules may apply in other countries.

A conflict between workers and managers in the post-pandemic drive to "return everyone to the office" has been reported in the popular media. It may be best to follow

an Agile approach: be flexible and work together as a team whether in-person, hybrid, or virtual. Agile workgroups see workplace issues that call for team-based solutions, including team members who are reluctant to return to the office [13], uncertainty about the effectiveness of hybrid work [14], and even concerns about new kinds of workplace surveillance [15]. Over time, we expect that organizations will adapt to meet the demands of a hybrid world.

References

1. Fraser, S., Mancl, D.: The Future of Conferences Research Survey (2022). <https://manclswx.com/survey2022.html>. Accessed 16 Jul 2022
2. Brooks, F.P.: The Mythical Man-Month. Addison-Wesley, Boston, MA (1995)
3. Conway, M.E.: How Do Committees Invent? Datamation (1968)
4. DeMarco, T., Lister, T.: Peopleware. Addison-Wesley, Boston, MA (2013)
5. Fraser, S. et al.: Retrospectives on Peopleware. In: ICSE Companion, pp. 21–24 IEEE Computer Society (2007). <https://doi.org/10.1109/ICSECOMPANION.2007.61>
6. Smite, D., Moe, N.B., Klotins, E., Gonzalez-Huerta, J.: From forced working-from-home to working-from-anywhere: two revolutions in telework (2021). <https://arxiv.org/abs/2101.08315>. Accessed 22 Jul 2022
7. Brucks, M.S., Levav, J.: Virtual communication curbs creative idea generation. *Nature* **605**, 108–112 (2022). <https://doi.org/10.1038/s41586-022-04643-y>
8. 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 (2022). <https://arxiv.org/abs/2203.11136>. Accessed 22 Jul 2022
9. Tang, J.: Understanding the telework experience of people with disabilities. *Proc. ACM Hum.-Comput. Interact.* **5**, 1–27 (2021). <https://doi.org/10.1145/3449104>
10. Goldberg, E.: A two-year, 50-million-person experiment in changing how we work. *New York Times*, March 10, 2022 (2022). <https://www.nytimes.com/2022/03/10/business/remote-work-office-life.html>. Accessed 24 Jul 2022
11. Boden, S.: Remote work is commonplace now, and workers with disabilities could benefit from the change. *WESA Radio Pittsburgh*, Apr. 3, 2022 (2022). <https://www.wesa.fm/health-science-tech/2022-04-05/remote-work-is-commonplace-now-and-workers-with-disabilities-stand-to-benefit>. Accessed 24 Jul 2022
12. Strickland, K.: Remote work as a reasonable accommodation: implications from the COVID-19 pandemic. *Harvard Civil Rights - Civil Liberties Law Review* (2021). <https://journals.law.harvard.edu/crcl/remote-work-as-a-reasonable-accommodation-implications-from-the-covid-19-pandemic/>. Accessed 24 Jul 2022
13. Hsu, A.: The idea of working in the office, all day, every day? No thanks, say workers. *NPR article* (2022). <https://www.npr.org/2022/06/05/1102744672/remote-work-from-home-return-to-office-covid-pandemic-workers-apple-google>. Accessed 16 Jul 2022
14. Christian, A.: Why hybrid work is emotionally exhausting. *BBC News article* (2022). <https://www.bbc.com/worklife/article/20220120-why-hybrid-work-is-emotionally-exhausting>. Accessed 16 Jul 2022
15. Christian, A.: The employee surveillance that fuels worker distrust. *BBC News article* (2022). <https://www.bbc.com/worklife/article/20220621-the-employee-surveillance-that-fuels-worker-distrust>. Accessed 16 Jul 2022

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.



Organisational Debt and Large-Scale Agile

A Summary of the First International Workshop on Organizational Debt and Large-Scale Agile Software Development

Tomas Gustavsson and Muhammad Ovais Ahmad

Karlstad University, Karlstad 651 88, Sweden
{tomas.gustavsson,muhammad.ovais.ahmad}@kau.se

Abstract. This is a summary of the First International Workshop on Organisational Debt and Large-Scale Agile Software Development. The workshop addressed organisational debt in agile software development through research presentations and discussion sessions. The keynote highlighted team interdependencies and the importance of coordination between teams. Research papers presented concepts on mental associations, methods to measure agile maturity, and the impact of organizational debt on coordination mechanisms.

Keywords: Organisational debt · Large-scale agile · Agile software development

1 Introduction

In software engineering, technical debt has been widely investigated, but debt regarding social issues, people, and processes has not been explored as much [1]. It should be noted here that we use organisational debt as an umbrella term to cover process debt, people debt, culture debt, social debt. There is limited and descriptive research on social debt and process debt [2], whereas almost no studies on organisational debt in software context [3]. Agile software development focuses on swiftly responding to change, constant deliveries and customer collaboration [4]. Such a swiftly value-driven demanding environment is a breeding ground for suboptimal processes that might have short-term benefits but an overall negative impact on the organization in the medium-long term. Organisational debt can incur either intentionally or unintentionally through management actions when short-term advantages are sought at the expense of quality and sustainability [5].

The goal of the workshop was to facilitate knowledge-sharing about organisational debt and deepen the knowledge about mitigation strategies and practices to manage organisational debt. To that end, the workshop was not only based on research presentations, but it also provided discussions among attendants.

2 Workshop Development

The workshop was divided into two sessions. In the initial session, Bas Vodde delivered a keynote entitled “Maximizing Dependencies with Interdependent Teams”, offering an insight into various dependency types and coordination facets. Vodde illuminated challenges in coordination, underscoring how team structures, whether component or feature-focused, can potentially contribute to organisational debt. This presentation sparked rigorous discussions surrounding team formation and the myriad challenges and solutions inherent to large-scale agile practices.

The second session contained paper presentations interspersed with group discussions. Ayan Chakraborty initiated with “Understanding Mental Associations and its Impact on Mindset of Scrum Teams”, posing that Scrum implementation challenges might be contingent upon team members’ mindsets. He probed if individual mental associations influenced the collective Scrum team mindset, uncovering discernible patterns linking team members’ mental associations to their overarching perspectives, potentially explaining resistance to mindset transitions.

Maarit Laanti followed with “Updated Agile Transformation Model for Large Product Development Organizations”. She delineated an evolved model for enterprises undergoing agile shifts. Given the burgeoning application of agile methodologies in hardware and product development, Laanti’s model was recalibrated according to the most recent insights on agile implementation within extensive product development structures.

The third presentation was “Organizational Debt in Large-Scale Hybrid Agile Software Development: A Case Study on Coordination Mechanisms” and was presented by Zixuan Liu. She presented a case study that identified organizational debt challenges such as a lack of shared mental models, team coordination, team cohesion, and team learning. The study showed how hybrid working arrangements created tension between increased individual autonomy and team objectives and between team autonomy and inter-team coordination.

After the presentations, workshop attendees were afforded the opportunity to engage directly with a presenter of their choice. These breakout sessions facilitated in-depth dialogues on the nuances, hurdles, and enhancements pertinent to each study. Consequently, the workshop transitioned beyond mere academic presentations on organizational debt and large-scale agile, fostering an environment of collaborative discourse among participants.

3 Workshop Conclusions

In reflecting upon the proceedings of the three-hour intensive workshop, it becomes palpable that the domain of organizational debt remains largely unexplored. While technical debt has garnered substantial scholarly attention over time, the multifaceted nuances of organizational debt need deeper exploration. The keynote and ensuing discussions underscored the challenges associated with team dependencies, formation, and coordination, each constituting potential sources of organizational debt.

The research presentations further accentuated the need to comprehend distinct types of organizational debt. For instance, discerning whether challenges in Scrum implementation arise from the mindset of team members or if the organizational structure serves as

a potential liability. Understanding the unique organizational challenges in these expansive contexts becomes paramount as agile methodologies penetrate broader domains like hardware and product development.





Future scholarly endeavors should prioritize amplifying rigorous empirical studies focusing on organizational debt within large-scale agile entities. There is a necessity to assess and quantify organizational debt's ramifications systematically. Only through such measured analyses can effective strategies be developed to anticipate, identify, and ultimately mitigate the underlying causes and consequences of this debt in the intricate tapestry of software development.

References

1. Ahmad, M.O., Gustavsson, T.: The pandora's box of social, process, and people debts in software engineering. *J. Softw Evol. Process* **e2516** (2022)
2. Lenarduzzi, V., Besker, T., Taibi, D., Martini, A., Fontana, F.A.: A systematic literature review on technical debt prioritization: strategies, processes, factors, and tools. *J. Syst. Softw.* **171** (2021)
3. Alfayez, R., Alwehaibi, W., Winn, R., Venson, E., Boehm, B.: A systematic literature review of technical debt prioritization. In: *Proceedings of the 3rd International Conference on Technical Debt*, pp. 1–10 (2020)
4. Dikert, K., Paasivaara, M., Lassenius, C.: Challenges and success factors for large-scale agile transformations: a systematic literature review. *J. Syst. Softw.* **119**, 87–108 (2016)
5. Klinger, T., Tarr, P., Wagstrom, P., Williams, C.: An enterprise perspective on technical debt. In *Proceedings of the 2nd Workshop on Managing Technical Debt* (2011)



Organizational Debt in Large-Scale Hybrid Agile Software Development: A Case Study on Coordination Mechanisms

Zixuan Liu¹ , Viktoria Stray^{1,2}  , and Tor Sporsem² 

¹ University of Oslo, 0373 Oslo, Norway
stray@ifi.uio.no

² SINTEF, Trondheim, Norway

Abstract. Software development is a complex human-centered activity, increasingly complicated by agile organizations scaling and adopting hybrid work. While technical debt has been extensively studied, other forms of debt-organizational, process, cultural, and social-have received less attention. We conducted a case study using ten semi-structured interviews, observations, and document analysis to identify coordination mechanisms used in large-scale hybrid agile. We identified organizational debt challenges such as a lack of shared mental models, team coordination, team cohesion, and team learning. Also, the hybrid working arrangement was found to create tension between increased individual autonomy and team objectives, as well as between team autonomy and inter-team coordination. We found 23 coordination mechanisms that the teams used to address challenges in their organization. We propose that implementing many of these mechanisms may help manage organizational debt.

Keywords: Agile transformation · Collaboration · Teamwork
Coordination strategies · Knowledge sharing · Scalability challenges

1 Introduction

As companies adjust to the post-pandemic work-life, managers have been grappling with whether and how to bring employees back to the office, and many companies offer a hybrid work solution. Many employees see work location flexibility as a bonus on par with increased salary [3]. However, many experience difficulties related to communication, collaboration, and cooperation with other team members [2, 10, 16] when some or all are working from home.

Managers risk creating organizational debt, such as process debt [15] and social debt [23], when making new policies for hybrid work. New policies influence what new norms are created among workers. If managers permit poor norms to take root, they may find themselves having to “pay off” this organizational debt in the future, making it crucial to implement effective policies

from the outset. Is hybrid work truly the “best of both worlds”, or is it rather the worst of both worlds? What will developers choose to do when given the freedom to choose between working from home and at the office? Several studies in software engineering have called for further research in the benefits, challenges and coordination strategies of distributed or hybrid software development teams [9, 16, 18, 19].

Technical Debt has been rigorously investigated [14], and the metaphor is now used to describe also other types of organizational debt. For example, the debt metaphor has also been used in requirements engineering [11]. Ahmad and Gustavsson [1] recently conducted a systematic mapping review on nontechnical debt in software engineering and found 17 studies that investigate social, process, and people debts. They reported that both [8, 17] found lack of communication, collaboration and coordination to be the cause of social debt. Lack of coordination is a common organizational challenge and has been found to be a cause of process debt [15].

Agile development at scale introduces new challenges. For example, there are more uncertainty, complexity, and dependencies between projects and teams, and thus coordination especially becomes a challenge [6]. Furthermore, the high degree of complexity and dependencies across teams threatens team autonomy [12]. A recent longitudinal study exploring coordination mechanisms in large-scale agile revealed that these mechanisms evolve in response to external and internal change events and that implementing the appropriate coordination mechanisms can significantly enhance the organization’s resilience [4]. We hereafter use the umbrella term *organizational debt* to include both social and process debt. We aimed to understand how challenges with coordination, that cause both process and social debt, can be managed by the use of coordination mechanisms. We explored the following research question:

RQ1: “*What coordination mechanisms are used to manage organizational debt challenges in large-scale agile?*”

2 Methodology

The research was carried out in the case organisation “*PubTrans*”, which is an organisation responsible for the software development project of a platform for public transportation in Norway. The project has existed since 2016, and the first author was hired as an IT consultant in the project from the end of 2019.

The company could be defined as *large-scale*, as it has seventeen development teams ranging between five and eighteen team members, each with their own responsibility area, and together working toward developing the same products. The teams are able to choose their own tools, technology, agile methods and processes, and could thus be considered autonomous. As the pandemic restrictions were lifted in the middle of 2021, the company decided to experiment with a *hybrid working arrangement*, as many of its employees enjoyed the flexibility of being able to work remotely. Teams at PubTrans were allowed to design their

own approach to the hybrid working arrangement, due to a high degree of autonomy. As a result, the teams continuously experimented with new coordination strategies throughout the pandemic and post-pandemic period.

We carried out and transcribed 10 semi-structured interviews with developers and designers, spanning 7 different development teams at PubTrans. Also, we collected artefacts which were relevant to the hybrid working arrangement, such as Slack logs and documentation from Jira, Confluence, Miro and Microsoft Teams. This was done in order to enable better data triangulation, and to improve the validity and reliability of the case study. The research started in January 2022, and an overview of the research timeline is illustrated in Fig. 1.

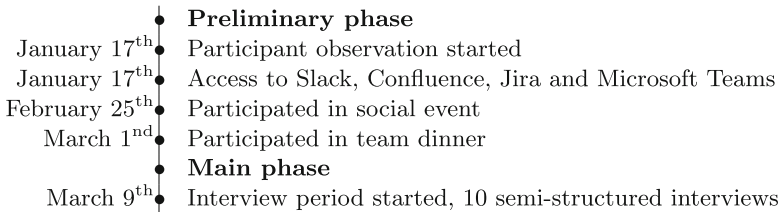


Fig. 1. Timeline of research

When choosing informants, we minimized sampling bias through *diversity* and *variation* as selection criteria. We initially compiled a list of all potential informants within PubTrans, encompassing about 100 individuals engaged in software development. Given diverse experiences across roles - developers, designers, team leads, and managers - a comprehensive exploration of all roles was impractical for this study's scope. Consequently, we focused on developers and designers at PubTrans, offering a diverse skillset within the informant group while maintaining manageable interview scope. In the preliminary research phase, it became evident that autonomous teams exhibited varying approaches to the hybrid work arrangement. To capture a spectrum of perspectives, we opted to select one or two representatives from several teams, facilitating the gathering of diverse narratives. Additionally, we deliberately chose informants with varying experience levels, personalities, life situations, and work settings (home or office).

A *thematic coding analysis* approach was taken when analyzing the data. We first created root nodes on NVIVO following the theoretical framework, and deductively generated the initial codes. However, we also inductively generated codes in order to stay close to the data. This resulted in a list of benefits and challenges caused by the hybrid working arrangement, and a list of coordination mechanisms identified based on the model by [22].

From January 2022 to March 2022, the first author assumed the role of a participant observer within the PubTrans organization. The participation included attending planned and impromptu meetings, formal discussions, informal interactions, and team collaborative efforts. Additionally, engagement extended to

social gatherings like Friday gatherings and seminars. These activities provided a comprehensive understanding of the intricacies inherent to the PubTrans company. This understanding was particularly profound during the period of hybrid work arrangements. Importantly, this active involvement facilitated the establishment of rapport with key individuals before the subsequent interview phase.

3 Results

3.1 Shared Mental Models

We found one major challenge to be maintaining a *shared mental model* of the location and availability of other team members within the team. First, many informants were unsure about how many of their coworkers will be at the office on any given workday. Most did not see a reason to travel to the PubTrans office when their collaborators were working remotely, and only wished to co-locate if enough team members were at the office. Some felt lonely when staying at the office without other team members, as they did not know many other employees. Many wished to adjust their co-location plans according to the other team members, but this was difficult as most team members did not document their co-location plans. The majority decided their work location right before the work day started. “Yesterday I went to work, and there was no one there, and that wasn’t so fun. I could’ve just as well stayed at home” (Interview C2). Likewise, it was even more difficult for the employees to find the co-location plans of other teams. Most employees informed about their co-location plans in private team Slack channels, which were hidden from those outside the team.

Secondly, the informants experienced difficulties *accommodating the other work mode* when working from separate locations, due to a lack of shared awareness. This created challenges related to *inclusivity*. For example, those who were co-located at the office could often forget to accommodate to those working remotely, and those alone at the office could experience difficulties trying to participate in all the digital activities in an open office landscape. As a result, information shared between the co-located team members may not reach those working remotely. Similar challenges were also reported from *hybrid meetings*. Those who were co-located sometimes had informal conversations which excluded the digital participants. These conversations could be disruptive to the remote participants, and the remote participants also missed out on important information and decisions.

3.2 Team Coordination

We identified *team coordination* as a major organizational debt challenge at PubTrans, with hybrid work increasing meeting complexity and communication barriers. The interviewees reported fewer ad-hoc meetings in hybrid teams due to reduced co-location time. Initiating video calls without prior planning on Slack was uncommon, unlike in-person interactions at the office. Ad-hoc meetings

mostly occurred when teams were co-located. “*It’s a bit more painful when you’re at home. You always have to call. It’s such a big step [...] it feels like you’re interrupting others way more. Like, ‘oh my god, now there’s another message or a direct phone call from him’. It feels so dramatic*” (Interview B1).

Secondly, the frequency of communication *between* teams was significantly reduced. Prior to the pandemic, teams working on similar domains at PubTrans were placed in the vicinity of each other, thus promoting the collaboration between these two teams. Similarly, task forces (i.e. a temporary team consisting of members from different teams working on the same feature) were built prior and during the pandemic as an inter-team coordination mechanism. This had however disappeared as a result of the hybrid work arrangement, as the teams no longer came to the office on the same days.

The employees also experienced *longer feedback loops* when collaborating on the same task, compared to before the pandemic. This was due to the increased barriers to initiating ad-hoc communication. In hybrid teams, conversations had to be more explicitly planned and executed using communication tools such as Slack and Microsoft Teams. Setting up the tools and waiting for asynchronous answers resulted in longer waiting time, and longer feedback loops during the collaboration sessions. This increased the threshold for asking questions, and decreased the coordination efficiency between team members.

3.3 Team Cohesion

Team cohesion was also identified as a major challenge caused by hybrid work. The team members reported to have decreased levels of attachment to the team, due to a lack of face-to-face social activities and informal conversations. It was more difficult to carry out informal conversations with the entire team in hybrid teams, as a result of the reduced and mismatched co-location time. Communication via tools such as Slack often felt impersonal, as they lacked additional dimensions such as body language. Furthermore, prior to the pandemic, the employees often gathered for dinners and other social activities after work. The frequency of social activities had drastically decreased after the pandemic.

Finally, despite encouragements to co-locate on particular days, some team members preferred to *never* come to the office. In comparison to individuals who frequently worked from the office, informants said it was harder to get to know the remote team members. Due to significant obstacles to initiating conversations when working remotely, the hybrid teams spent more time on casual interactions when co-located than before the pandemic. Those who preferred to not co-locate were thus excluded from these interactions.

3.4 Team Learning

At PubTrans, *team learning* was the fourth major challenge in hybrid teams. Due to the limited and mismatched co-location time, many people had difficulty asking questions and transferring knowledge. Particularly, the sharing of *domain*

knowledge and *tacit knowledge* was cited by almost every informant as a significant challenge. The project’s large scope necessitated the integration of a vast number of teams, subsystems, stakeholders, and technology. Understanding the PubTrans domain therefore required an understanding of the organisation as a whole, and this knowledge was frequently tacit and undocumented. Almost all of the informants said that obtaining domain knowledge was more challenging than learning specific technologies and programming languages. While there were many internet resources for specific programming languages, finding answers to inquiries about the PubTrans domain was impossible.

Table 1. Coordination mechanisms managing organizational debt

		Shared mental models	Team coordination	Team cohesion	Team learning
Co-location structure mechanism	Co-location with team				
	Co-location across teams				
	Co-location rules				
	Incentives for co-location				
Slack synchronization & boundary- spanning tool	Slack as main coordination tool				
	“Good morning” message on Slack				
	Slack icons				
	Inter-team Slack channels				
Meetings synchronization & boundary- spanning activity	Daily standup				
	Retrospective meeting				
	One-on-one meeting				
	Informal conversations during meeting				
Tools for hybrid meetings Synchronization & boundary- spanning tool	Video conferencing software				
	Digital calendars				
	Integrated meeting rooms				
Documentation Synchronization & boundary- spanning tool	Documentation				
	Documentation tools (Jira, Confluence)				
	Visual collaboration tools (Miro)				
Gaming Synchronization & boundary- spanning activity	Multiplayer games				
	Quizzes				
Pair-programming Synchronization activity	Pair-programming				
Physical social events Synchronization & boundary- spanning activity	Social events with team, e.g. team dinner				
	Physical seminars				

4 Discussion

We will now discuss our research question: “What coordination mechanisms are used to manage organizational debt challenges in large-scale agile?” In large-scale

agile environments, agile practices are often used together with other organizational practices [5]. This was reflected in our findings, as the coordination mechanisms identified included both agile coordination practices, such as stand-up meetings, retrospective meetings and task boards, as well as non-agile practices like social events and gaming.

In total, we found 23 coordination mechanisms used to manage organizational debt challenges. We have categorized them into eight general coordination mechanisms and if they were affecting the aspects: Shared mental models, team coordination, team cohesion and team learning, see Table 1. Some coordination mechanisms are closely interrelated - for instance, creating shared mental models also improves the team cohesion, which in turn encourages team learning.

Shared mental models represent knowledge held in common by members that lets them understand tasks and relationship among tasks, and coordinate their actions and interactions [7]. Our findings suggested that hybrid teams should *explicitly discuss their hybrid work processes*, in order to create a shared mental model amongst its members. This is in line with other research [18, 21].

The coordination mechanisms *co-location rules* and *Slack norms and etiquette*, helped by explicitly discussing and describing the hybrid collaboration pattern within the teams. Discussing common *co-location rules* created a shared understanding of the team's *work location* on a given work day. The team members could thus anticipate one another's needs, and adjust their co-location plans accordingly. Our work builds on the findings that Slack is an essential tool for coordination in distributed teams [20]. Further, in our teams, the shared *Slack etiquette* created a common understanding of the *availability* of team members, by for example agreeing on traditions such as sending "good morning" messages when ready for work, and changing Slack icons when unavailable.

We found a set of coordination mechanisms that teams might utilize to increase *team cohesion*, which could improve the overall team performance [7]. These mechanisms included *social activities with a common purpose*, such as multiplayer games, daily quizzes, physical social events with the team and the company, and informal conversations during meetings. We found coordination mechanisms employed to encourage *team learning*. These mechanisms generally focused on *lowering barriers to asking questions* within and across teams. Frequent communication within teams are shown to improve psychological safety, which in turn lowers the barriers to asking questions and encourages knowledge sharing [4, 13].

In our study, teams worked in a hybrid setting. Unlike co-located large-scale agile, hybrid organizations must align co-location rules across teams to establish a shared mental model organization-wide. To ease tension between team autonomy and inter-team alignment, creating incentives for co-location can be beneficial. Enforcing company-wide rules, such as co-location on a set day, may threaten team autonomy in a large-scale agile organization [12]. Instead, *incentives* could be used to encourage co-location. For instance, PubTrans introduced the "baked goods trolley" on Thursdays, strengthening the employees' willingness for co-location.

5 Conclusion

In conclusion, our research highlights the importance of addressing not only technical debt but also other types of organizational debt, such as process and social debt, in software development organizations, especially in the context of large-scale hybrid settings. We identified 23 coordination mechanisms that were used to manage organizational debt. By raising awareness of these non-technical forms of debt and the coordination mechanisms to address them, we aim to provide practitioners with insights to improve their software development processes and enhance overall organizational performance. Further research is encouraged to uncover additional strategies that could be beneficial in managing the complexities of large-scale, hybrid agile organizations.

Acknowledgements. Thanks to the company for their research engagement. This work was supported in part by the Research Council of Norway (Project Transformit, grant 321477).

References

1. Ahmad, M.O., Gustavsson, T.: The pandora’s box of social, process, and people debts in software engineering. *J. Softw. Evol. Process* (2022)
2. Bao, L., Li, T., Xia, X., Zhu, K., Li, H., Yang, X.: How does working from home affect developer productivity?-a case study of baidu during the covid-19 pandemic. *Sci. China Inf. Sci.* **65**(4), 142102 (2022)
3. Barrero, J.M., Bloom, N., Davis, S.J.: Let me work from home, or i will find another job. University of Chicago, Becker Friedman Institute for Economics Working Paper (2021–87) (2021)
4. Berntzen, M., Stray, V., Moe, N.B., Hoda, R.: Responding to change over time: A longitudinal case study on changes in coordination mechanisms in large-scale agile. *Empirical Softw. Eng.* **28**(114) (2023)
5. Berntzen, M., Stray, V., Moe, N.B.: Coordination strategies: managing inter-team coordination challenges in large-scale agile. In: Gregory, P., Lassenius, C., Wang, X., Kruchten, P. (eds.) *XP 2021. LNBIP*, vol. 419, pp. 140–156. Springer, Cham (2021). https://doi.org/10.1007/978-3-030-78098-2_9
6. Dingsøy, T., Bjørnson, F.O., Schrof, J., Sporse, T.: A longitudinal explanatory case study of coordination in a very large development programme: the impact of transitioning from a first- to a second-generation large-scale agile development method. *Empir. Softw. Eng.* **28**(1), 1 (2023)
7. Dingsøy, T., Fægri, T.E., Dybå, T., Haugset, B., Lindsjörn, Y.: Team performance in software development: research results versus agile principles. *IEEE Softw.* **33**(4), 106–110 (2016)
8. Dreesen, T., Hennel, P., Rosenkranz, C., Kude, T.: “The second vice is lying, the first is running into debt.” antecedents and mitigating practices of social debt: An exploratory study in distributed software development teams (2021)
9. Florea, R., Stray, V.: A global view on the hard skills and testing tools in software testing. In: 2019 ACM/IEEE 14th International Conference on Global Software Engineering (ICGSE), pp. 143–151. IEEE (2019)

10. Ford, D., et al.: A tale of two cities: software developers working from home during the covid-19 pandemic. *ACM Trans. Softw. Eng. Methodol. (TOSEM)* **31**(2), 1–37 (2021)
11. Frattini, J., et al.: An initial theory to understand and manage requirements engineering debt in practice. *Information and Software Technology* (2023)
12. Gustavsson, T., Berntzen, M., Stray, V.: Changes to team autonomy in large-scale software development: a multiple case study of scaled agile framework (safe) implementations. *Int. J. Inf. Syst. Proj. Manag.* **10**(1), 29–46 (2022)
13. Hennel, P., Rosenkranz, C.: Investigating the “socio” in socio-technical development: the case for psychological safety in agile information systems development. *Project Manage. J.* **52**(1), 11–30 (2021)
14. Lenarduzzi, V., Besker, T., Taibi, D., Martini, A., Fontana, F.A.: A systematic literature review on technical debt prioritization: strategies, processes, factors, and tools. *J. Syst. Softw.* **171**, 110827 (2021)
15. Martini, A., Stray, V., Moe, N.B.: Technical-, social- and process debt in large-scale agile: an exploratory case-study. In: Hoda, R. (ed.) *XP 2019. LNBP*, vol. 364, pp. 112–119. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-30126-2_14
16. Nolan, A., White, R., Soomro, M., Dopamu, B.C., Yilmaz, M., Solan, D., Clarke, P.: To Work from Home (WFH) or not to work from home? lessons learned by software engineers during the COVID-19 pandemic. In: Yilmaz, M., Clarke, P., Messnarz, R., Reiner, M. (eds.) *EuroSPI 2021. CCIS*, vol. 1442, pp. 14–33. Springer, Cham (2021). https://doi.org/10.1007/978-3-030-85521-5_2
17. Ramač, R., et al.: Prevalence, common causes and effects of technical debt: results from a family of surveys with the it industry. *J. Syst. Softw.* **184**, 111114 (2022)
18. 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)
19. Sporsem, T., Moe, N.B.: Coordination strategies when working from anywhere: a case study of two agile teams. In: *Proceedings of the 23rd International Conference on Agile Software Development, XP 2022*, 2022, pp. 52–61. Springer (2022)
20. Stray, V., Moe, N.B., Vedal, H., Berntzen, M.: Using Objectives and Key Results (OKRs) and Slack: a case study of coordination in large-scale distributed agile. In: *Proceedings of the 55th Hawaii International Conference on System Sciences*, p. 10 pages. HICSS (2021). <http://hdl.handle.net/10125/80225>
21. Strode, D., Dingsøyr, T., Lindsjorn, Y.: A teamwork effectiveness model for agile software development. *Empir. Softw. Eng.* **27**(2), 1–50 (2022)
22. Strode, D., Huff, S.L., Hope, B., Link, S.: Coordination in co-located agile software development projects. *J. Syst. Softw.* **85**(6), 1222–1238 (2012)
23. Tamburri, D.A., Kruchten, P., Lago, P., van Vliet, H.: What is social debt in software engineering? In: *2013 6th International Workshop on Cooperative and Human Aspects of Software Engineering (CHASE)*, pp. 93–96. IEEE (2013)

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.



Software-Intensive Business



The Know-How of Agile Retrospectives in Software Startups

Dron Khanna^(✉) and Xiaofeng Wang

Free University of Bolzano, 39100, Bolzano, Italy
{dron.khanna,xiaofeng.wang}@unibz.it

Abstract. Software startups are responsible for fast product delivery to the market. To aid this process, a retrospective inside a startup team can be very fruitful for software development. The traditional way of conducting agile retrospectives involves discussion based on what went well, what did not go well, and how to improve the software development cycle helps to save resources, get directed toward the startup vision, and overcome several challenges. To attain insights about the agile retrospective approach in startups, we studied the following question: How are software startups performing agile retrospectives? Hence, we conducted seven multiple case studies with 19 semi-structured interviews that lasted 30–65 min. The results outline that all software startups prefer a reflection through agile retrospectives but not in the traditional manner. Due to the startup's casual and less restricted working environment, teams prefer informal agile retrospectives, which involve no confined boundaries of time, venue, and participants.

Keywords: Traditional Retrospective · Informal Retrospective · Team Retrospective · Agile Retrospective · Continuous Retrospective

1 Introduction

Software startups tend to be known for their casual and less restricted working environment than large organizations. Teams intended to work within a startup are less confined to working hours than 9:00 am–5:00 pm. The co-founders prefer to work at their own pace [8]. In the early days of a startup, often, co-founders are not fully-time devoted to the business idea. Partially they might be involved outside the startup life cycle [16].

Startups following agile practices to run the business idea go for agile retrospectives to improve the software development life cycle. The traditional way that encouraged startup teams involves basic questions; what went well in the group? What did not go well? Moreover, what to improve during the agile retrospectives [1,2]. It involves five stages in agile retrospectives; set the stage, gather data, generate insights, decide what to do, and close the retrospectives [2]. Despite that, agile retrospectives lack crucial components such as; which

team members should be involved during the practice. Where should the retrospective be conducted? Moreover, What could be the correct duration of the retrospective session? Several studies have mentioned the benefits of agile retrospectives, such as saving startup resources, enlarging teams, obtaining and getting directed toward the vision, and overcoming many challenges [3]. However, still, the startup failure stories weigh higher than successful ones. Hence, we feel that other studies need the view of agile retrospectives that contribute to assisting the software startup journey. Many startup teams no doubt ask if we are doing the retrospective correctly [4]. Could it be done in another manner? Is it finished with this iteration, or should it be continued? Moreover, if so, to what extent? Considering the various open venues, we formulate the following research question (RQ) that this study seeks to tackle: **How are software startups performing agile retrospectives?**

The rest study is as follows. The literature review section discusses the existing literature and theoretical framework related to the agile retrospective. Presented in Sect. 3 is the research methodology, which describes the multiple case study conducted with seven software startups. Then we describe the findings of the study in the next section. Section 5 discusses these results from existing literature, and finally, Sect. 6 concludes the paper.

2 Literature Review

“Retrospective” is a practice that deals with reflection thoughts that emerge in the brain [5] of team participants. Retrospectives originated from the Latin word “retrospectare” meaning looking back in the past to the thoughts. Retrospectives occur on experiences with tentative dates or periods to trace back the event [6]. A common retrospective practice is called “Agile retrospectives” [6]. Agile retrospectives come under the roof of agile methods such as programming, daily stand-up meeting, and short iteration. These meetings occur at the regular onset of timely sprints [7]. A team does an agile retrospective at the beginning of iterations. A study defines an agile retrospective as a lessons-learned meeting [1]. As a team-driven practice, the team reflects on how the iteration went and how team members can improve future iterations. It also leads to various changes in the software development cycle [8]. The commonly applied method involves the five stages, which take nearly 35–45 min to complete the agile retrospective. The below list describes the five stages [2].

1. Set the stage - preparing the team with warm-up or ice-breaking activities.
2. Gather Data - sharing the experience of the topic that has to discuss in the retrospective.
3. Generate Insights - Collecting problems, success stories and failures
4. Decide what to do - Based on the previous step, prioritize the things to do for the next iteration.
5. Close the retrospective - Appreciate the team for accomplishments and interactions that help them reflect during the retrospective.

Often startups alter retrospectives practices. For example, in a startup case study, the team conducted a retrospective after a two-week sprint. After the end of the bi-weekly sprint, the team followed a specific activity addressing the problems in the previous retrospective. To identify the problems, the team structured an interview with the scrum master [9]. In another example, the student startup team reflects through agile retrospective practice after sixteen weeks. The startup team performed the retrospective in the seventeenth week, where the discussion yielded what the team learned. [10]. Hence, if we compare these two mentioned cases, we observe different time gaps that teams used to conduct the retrospective. Also, we observe from the case maturity level that the startup is, as the student teams were involved in the last startup. The teams had several iterations before they conducted the agile retrospectives. The retrospective should include a reflection on the experience that the participant earned during the startup development cycle. A team should conduct an agile retrospective to improve the overall startup system so the members involved can work easily and benefit the startup [3].

2.1 Theoretical Framework

HOW and WHAT? Verbal communication, writing notes, asking questions and discussions	WHO ? Two or more than two people
WHEN ? Working hours, spontaneous time	WHERE ? Office and outside the office

Fig. 1. Four theoretical blocks that leads to retrospective

A team could practice various techniques to reflect on experience [11] during the retrospectives [12]. Figure 1 shows four theoretical blocks from the literature mentioned below that constitute the framework. **How & What:** Talking and writing are some of the techniques used by various verbal and written reflection notes during the meeting. Teams could discuss and ask questions with each other to enhance the learning involved in the retrospective [11]. **Who:** Two or more than two members exist in the team during the retrospective session. When two team members are involved, each other often tends to help with reflection on the experience. More than two members, usually participants outside the team, could participate in the reflection approach [13] to do the retrospective [12]. **When:** Team members can perform reflection spontaneous time. This particular way of reflection gets encouraged without being specific to time boundaries [11, 14]. Teams can also be formally conducted during working hours [13, 14]. **Where:**

A team can perform informally or formally a reflection session. For example, formal is in the office, whereas informal sessions, team members can conduct outside the office [13,14].

3 Research Methodology

In this section, we represent the case description of 7 software startup cases. We conducted a multiple-case study with seven software startups. Multiple cases are selected to explore and diversify the breadth of the research question, and it allows making cross-comparison [15]. All the cases were founded in Italy that involved software development, either an application, web platform, or web-service provider. The startup's age range is between 2–5 years, with 3–5 members constituting the startup. We mainly involved the co-founders and managers during the interview because they were on the entire startup's journey. Most co-founders and managers have been interviewed twice. We conducted 19 semi-structured interviews. The duration of interview lasted between 30–65 min. We asked many background questions to ensure that startups have involved software development as the core that supports the business model. We transcribed all the interviews & later used Nvivo (a qualitative data analysis software package) to code. The team is the unit of analysis.

- **Case A** Startup is a mobile platform that enables the execution of crowd-sourcing campaigns with users carrying mobile devices.
- **Case B** is a software development kit provider. A customized platform offering marketing services to companies that wish to reach customers.
- **Case C** is a security and commercial device provider to customers. It operates in intelligent mobility and has developed a georeferenced digital communication system.
- **Case D** is a platform connecting different outdoor experiences. It helps to connect people who like outdoor experiences and certified guides.
- **Case E** is an intelligence application for indoor spaces. It specializes in delivering value-added services starting from tracking indoor objects.
- **Case F** is a sports platform. It provides services created to enhance synergies among work colleagues if they lack social relations that well-being services could complete.
- **Case G** focuses on designing and developing data analytic solutions for various industries. The startup helps other industries manage the data and make it more straightforward with immediate reports.

4 Findings

Table 1 highlights how the startup teams prefer to do retrospectives. We found that software startups are highly inclined towards an informal agile retrospective approach involving participant reflection. The teams participating in the retrospective do not have (time or office working) space boundaries. It could

Table 1. How are software startups performing agile retrospectives?

How and What teams do?	Who are involved in the team?	What is the usual duration?	Where does the team do the retrospectives?
Talking	All members	5–20 min	Coffee machine, meeting or empty room in office
Writing	Customer	30–45 min	Walking in nature
Multimedia	Shareholders	2 h	Walking next to river
Asking questions	Only co-founders	Half day	Hiking on mountains
Feedbacks		Several hours	Sitting in park with team
Discussion			Taking team for Ski
Guidance			Taking team for Cycling
Reporting			Restaurant or bar
Storytelling			Empty apartment

be spontaneous or planned but not 100% round tables with whiteboards. Agile retrospectives do not occur systematically and linearly as startups proceed in their entrepreneurial journey with various transitory phases such as the early idea stage, team formation, hypothesis testing, and creating the minimum viable product. Crucial is to perform retrospectives in these phases rather than in the proposed or structured manner. The understanding of retrospectives to all the startups is inclined towards reflection and sharing the experiences to learn and improve, irrespective of time and venue.

4.1 How and What Do Startup Teams Do During the Retrospectives?

To understand how & what, we found out that startups prefer techniques such as talking, writing, and sometimes the use of multimedia during the retrospective. Talking is the desired method that helps exchange the mind's reflection thoughts. Participants are reflecting on the experiences and events that occurred exchange by sharing the communication “*We discussed. How to know the customer? (case A), We share our thoughts and discuss. Then speak out together and again think about it. What could we do differently? What was the cause? Only by speaking out others will hear it (case B)*”. The startup teams commonly practice feedback & discussions during the retrospectives sessions, which also helps to give a better future perspective for the startup journey “*Ask for feedback and share the product, business model, and business pitches. We discussed the feedback about the software development during the retrospective (case B)*”. Further, we also found that writing is another way of exchanging reflection thoughts. Teams do report and often prefer to write the experiences and share them in the form of stories with other participants during the retrospective. Handwritten notes such as Post-it notes guide participants to design better future workflow “*How did it*

happen with our team, good? Bad? What happened? Yes, through notes (case C), Think and let us write it down. It should be personal and face-to-face, where the team writes down (case E). If a team member does not want to talk about his story and hears everyone is telling their story, he opens up (case F)". Startups also mentioned using multimedia, such as presentation slides and videos, that moderates the reflection and sharing of experiences during the retrospective sessions *"We have people, and not everyone is in the same place, and some work from home. So, we do a video call and present (case C). We also use a personal chat or web chat (case F)"*.

4.2 Who Is Involved from the Team During the Retrospectives?

Startups mentioned it is majorly the team *"We started to think and discuss among the team. Everyone must share (case A)"* or in another scenario it is just the core team which is just the co-founders who are involved during the retrospective sessions *"Just in the founding team and we agree, so founders let us meet and discuss the issue (case G)"*. Depending on how critical the agenda is, participants are involved. Shareholders also participate during retrospectives *"Startup's shareholders sitting. Yes, we have shareholders in the meeting (case A)"*. On the other hand, one Startup did mention that in their early idea stage, twice they involved customers to get a better understanding of the problem *"With the core people and the other people. Sometimes when we discuss with our customers or partners, we do retrospective (case C)"*.

4.3 What Is the Usual Duration of the Retrospectives?

Startups mentioned different time duration depending on the importance and criticality of the topic. In difficult situations, the retrospective extends to an extensive period. The CEO even said that it took them half-day, and the shareholders repeated two times on different days because it was a point where a startup was about to get shut down *"When at some point, the startup was closing down. It was a retrospective, psychology speaking very hard, where all stakes in or not, black or white. We were asking What are we doing next? Are we closing down? Should we give it another try? That particular retrospective lasted one whole afternoon (case A)"*. Later after the critical retrospective, the startup involved the entire team in the upcoming session. Whereas some startups immediately discuss when problems come during the startup journey. This method could solve things quickly and last less than 20 min *"We are a small team and just 2 in the office. Our other members are working remotely. Whenever we get a problem, we turn our chairs around and talk. We do not wait for the end of the week" (case B)*. Talking with the teams as the problem exists could lead to multiple or continuous times of retrospectives. The problem can be the trigger for the continuous session. Some startups also mentioned teams were motivated by a prolonged session ranging from 30 min to 2 h *"Everyone from the team spends 5–20 min. Then we also discuss it again (case F). We do it for 30–45 min (case D). Two hours of reflection (case B)"*.

4.4 Where Does the Team Do the Retrospectives?

Mostly the startup prefers an informal venue that is outside the working space. They often conduct in the office, but being in the meeting room is not mandatory. Some teams prefer to go out and walk in nature, such as a mountain, river, or a park and reflect *“We do offsite. It would be best to sit somewhere outside the workplace. We try to do it most informally. We go to the mountains, and do some sports like cycling. Doing this offsite retrospective help (case E)”*. Also, startups like to switch places depending on the team’s preferences. Some members prefer to reflect while enjoying their hobby or sports such as jogging, skiing, or cycling *“We go to the mountains for reflection and doing hobby which the team would like (case G)”*. One startup at a critical stage conducted the retrospective at the apartment. It was a space that was not in use. The co-founders also mentioned that they were all sitting at some point, where they made prolonged eye contact too *“In a cold empty apartment. We needed a kind of safe house. We could cut from the rest of the world and discuss among ourselves. There was a tough retrospective meeting where we sat in a quiet apartment (case A)”*. Sometimes when team members meet at the coffee machine, do a small duration retrospective session *“Randomly when we meet at the coffee machine, and then we start discussing the long hold issues which turn into good reflection period (case G)”*. Young startup entrepreneurs also prefer to go to the bar after work and do the retrospective session *“Yes, go to a bar with the other co-founder, and we do some reflection or retrospective to discuss the startup problems (case D)”*. Finally, when the team works remotely, they prefer online retrospective meetings *We do retrospectives through video calls and without meeting up (case D). We do both, with meet up and doing some online with other team members. When issues can be severe (case B)”*.

5 Discussion

Software startups come across entrepreneurial experiences every day and many thoughts emerge in mind *“There are different things to do every day, which leads to experience (case E). Thoughts that need to be discussed during the retrospective (case F)”*. Figure 2 is the word cloud of the most repeated ways of doing the retrospective. Asking questions and discussing is also mentioned in the literature as a common way that enhances reflection in retrospectives [11]. The topic is the source to initiate retrospectives and lead a discussion *“Everyone can share the experience. Sometimes with co-founders and other colleagues (case F)”*. Compared to findings in the literature, an important thing to discuss is the criticality of the topic or problem conducted during the retrospective. Depending upon the case, the duration, the venue, and team members play the role during the retrospective session. The topic is the one for software startups that could moderate the reflection session during the retrospectives. The topic could make a startup retrospective very informal or formal. At the same time, the literature highlights a traditional approach of agile retrospective [1] with the 5-stage process, completed on average in 35–45 min [2]. From the case of [10] mentioned in



Fig. 2. Word cloud of techniques that startup teams do to conduct retrospective

the literature, the study was conducted only once in the retrospective without saying the time duration. This case is more corresponding to point (*case D*) where young startup entrepreneurs are more inclined to perform informal retrospectives. We can deduce from here that the more youthful a startup is, the more casual the retrospective approach is without any time duration, members involved, and venue to perform. Whereas with the startup life cycle, the more mature it gets the teams to start getting into boundaries of the time limit and people involvement. From Fig. 2, we can see the startup members are primarily involved, which could lead to various time duration. However, for the venue to conduct retrospectives, even mature startups mentioned in Sect. 3 prefer to do informal places and outside offices. The reason might be because reflection during retrospectives is susceptible outside a daily working environment [16].

6 Conclusion

The agile retrospective is a fruitful practice that helps software startups in many aspects and saves them from many dangerous situations. The traditional way involves five stages with three basic questions. Still, it lacks the know-how with various outlooks, such as the team member involvement, where it should be conducted. Hence, to understand better agile retrospectives regarding software startups, we formulated the questions: How are software startups performing agile retrospectives? We conducted multiple case studies with seven software startups, with 19 semi-structured interviews. We discovered that majorly startups do not prefer to work agile retrospectives formally. As startups are young institutions with starting points of the business idea [8], the approach is inclined to informal

agile retrospective. The practice does occur in the startup journey with a more flexible time duration and venue for the retrospective. Team members also vary depending on the criticality of the topic. At first, a retrospective involves the co-founders when the issue is censorious and, later, with the team. However, it is crucial to perform retrospectives continuously to obtain team learning which helps to grow a startup.

References

1. Gonçalves, L., Linders, B.: Getting value out of Agile retrospectives. A toolbox of retrospective exercises (2013)
2. Derby, E., Larsen, D., Schwaber, K.: Agile retrospectives: making good teams great. Pragmatic Bookshelf (2006)
3. Khanna, D., Wang, X.: Are your online agile retrospectives psychologically safe? the usage of online tools. In: tray, V., Stol, K.J., Paasivaara, M., Kruchten, P. (eds.) Agile Processes in Software Engineering and Extreme Programming: 23rd International Conference on Agile Software Development, XP 2022, Copenhagen, Denmark, 13–17 June 2022, Proceedings, pp. 35–51. Springer, Cham (2022). https://doi.org/10.1007/978-3-031-08169-9_3
4. Przybyłek, A., Kotecka, D.: Making agile retrospectives more awesome. In: 2017 Federated Conference on Computer Science and Information Systems (FedCSIS), pp. 1211–1216. IEEE (2017)
5. Brevig, L.: Engaging in retrospective reflection. *Read. Teach.* **59**(6), 522–530 (2006)
6. Khanna, D.: Experiential team learning in software startups. In: Proceedings of the 19th International Conference on Agile Software Development: Companion, pp. 1–3 (2018)
7. Yadav, V., Adya, M., Nath, D., Sridhar, V.: Investigating an ‘agile-rigid’ approach in globally distributed requirements analysis. In: PACIS 2007 Proceedings, p. 12 (2007)
8. Khanna, D., Nguyen-Duc, A., Wang, X.: From MVPs to pivots: a hypothesis-driven journey of two software startups. In: Wnuk, K., Brinkkemper, S. (eds.) ICSOB 2018. LNBIP, vol. 336, pp. 172–186. Springer, Cham (2018). https://doi.org/10.1007/978-3-030-04840-2_12
9. Matthies, C., Dobrigkeit, F., Ernst, A.: Counteracting agile retrospective problems with retrospective activities. In: Walker, A., O’Connor, R.V., Messnarz, R. (eds.) EuroSPI 2019. CCIS, vol. 1060, pp. 532–545. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-28005-5_41
10. Järvi, A., Taajamaa, V., Hyrynsalmi, S.: Lean software startup – an experience report from an entrepreneurial software business course. In: Fernandes, J.M., Machado, R.J., Wnuk, K. (eds.) ICSOB 2015. LNBIP, vol. 210, pp. 230–244. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-19593-3_21
11. Collier, P.J., Williams, D.R.: Reflection in action. *Learning through serving: a student guidebook for service-learning across the disciplines*, vol. 50, pp. 83–97 (2005)
12. Talby, D., Hazzan, O., Dubinsky, Y., Keren, A.: Reflections on reflection in agile software development. In: AGILE 2006 (AGILE 2006), p. 11. IEEE (2006)
13. Daudelin, M.W.: Learning from experience through reflection. *Org. Dyn.* **24**(3), 36–48 (1996)

14. Boud, D., Keogh, R., Walker, D.: *Reflection: Turning Experience into Learning*. Routledge, Abingdon (2013)
15. Yin, R.K.: *Case Study Research: Design and Methods*, vol. 5. Sage, Thousand Oaks (2009)
16. Khanna, D., Wang, X.: How software startup teams reflect: approaches, triggers and challenges. In: Hyrynsalmi, S., Suoranta, M., Nguyen-Duc, A., Tyrväinen, P., Abrahamsson, P. (eds.) *ICSOB 2019. LNBIP*, vol. 370, pp. 353–368. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-33742-1_28

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.





Analytics Practices in Practice: How Software Startup Companies Are Applying Analytics?

Usman Rafiq¹, Xiaofeng Wang¹, and Luciana Zaina²

¹ Free University of Bozen-Bolzano, Bolzano 39100,, Italy
{urafiq,xiaofeng.wang}@unibz.it

² Federal University of São Carlo (UFSCar), Sorocaba, Brazil
lzaina@ufscar.br

Abstract. Analytics is becoming known for providing exceptional opportunities to companies. However, companies are not utilizing its full potential despite the widespread benefits. In particular, analytics practices are imperceptible when we talk about small yet innovative companies like software startups. Nevertheless, startups' uncertain nature and focus on innovation make them promising candidates to increase the odds of success using analytics. In this paper, we investigate the key practices applied by software startups while conducting analytics. We address our research question through a multiple case study performed with three startup companies at distinct startup stages. Thematic data analysis led us to observe nine analytics practices applied by these startups. The identified analytics practices span a spectrum of activities characterized by the analytics framework. Our findings provide insights for software startup companies to introduce or enhance analytics in their specialized context.

Keywords: software start-up · tech firm · digital firm · data analytics

1 Introduction

Over the past few years, we have seen that analytics is becoming known for providing unprecedented opportunities to companies. Identifying customer needs, changing market segments, enhancing user experiences, optimizing processes, and getting competitive advantages are among a few potential domains where analytics can offer promising advantages to grow in the market [4, 6]. However, despite the hype and potential benefits of analytics, mentioned in both scientific and practitioner-oriented literature (e.g. [5, 12]), companies are not seen utilizing the full potential of it. For instance, according to survey results presented in a recent study [4], only 33% of established companies, 19% of medium companies, and 10% of small companies, from Europe, are using analytics. These statistics show that analytics has yet to become mainstream, particularly, for small companies.

Analytics practices inside small companies like software startups are imperceptible. However, the uncertain nature of startups and their focus on innovation

within limited resources make them promising candidates to practice it. Prior research concludes that startups realize the benefits of analytics [11] and analytics practices should be present as a toolkit to support startup operations in cutting operational costs, supporting decisions, and getting insights [7]. On the contrary, the existing literature sheds no light on the analytics practices currently employed by these companies. For example, a recent study [10] argues that best practices of applying analytics in a startup context are yet a mystery. Going in the same vein, when we particularly focus on software startup literature, we identify the challenges to analytics adoption faced by software startups [8] and the perception of startups about analytics [9]. In particular, the latest work on analytics in software startups [9], comprehends how analytics is understood in the startup context and what are the possible usage scenarios. Therefore, nevertheless, we lack in getting the key analytics practices applied by software startups while carrying out analytics-related operations. This is the research gap that our current study aims to address. Thus, the overall guiding question is:

RQ: How are software startups applying analytics?

Applying a case study research method [3], we gathered the data from three software startups at different startup life-cycle stages [1]. We primarily used semi-structured interviews to collect the data. Later, we utilized thematic analysis to analyze the collected data. The findings contribute in terms of providing insights for software startup companies to introduce or enhance analytics in their specialized context to unlock opportunities. The use of these practices will help them to avoid numerous pitfalls.

2 Analytics and Software Startups

Analytics has brought a significant transformation for several companies [16]. However, most companies are still striving to figure out the fundamentals of applying analytics in their environments [18]. While there is no consensus regarding the definition of analytics, most of the literature we came across (e.g. [5, 18]) refers to the definition offered by Davenport, Harris, and Morison [16]. According to the authors, analytics can be defined as “the use of analysis, data, and systematic reasoning”. In the startup context, we came across the definition provided by Croll and Yoskovitz [17], who think that analytics is to identify, measure, and track the riskiest parts of the startup. Thus, it measures and enlightens the way to product and market growth. The immediate outcome of practicing analytics is on the decision-making of the companies, which is transformed by insights generated from data [5]. Thus, analytics includes a set of tools, techniques [4] and human sense-making [5] to create value from data. In addition, utilizing analytics is neither a mechanical process [4] nor all the analytics practices can fit well in all sorts of companies [15].

Software startups are young and innovative companies that are characterized by limited resources, and dynamic markets and have to face multiple influences [14]. These companies offer software-intensive products or services and work

under extreme conditions of uncertainty [13]. While the research on software startups is evolving, numerous studies are focused on supporting engineering activities in these companies. Therefore, there are relatively few studies reporting on analytics for software startups. For example, two recent studies [8,9] discuss analytics in software startups. Together these studies report challenges that software startups face and comprehend how software startups understand analytics. In contrast, Berge et al. [11] studied hardware startups and explored benefits and challenges while utilizing analytics.

3 Research Method

We followed a multiple case study design in our study. According to Yin [3], case studies are suitable to address exploratory studies. In the same way, we studied multiple cases for a rich understanding of the key analytics practices. We approached 10 startups through our personal network, LinkedIn¹, and Crunchbase². Finally, three software startups agreed to participate in the research. We classified these startup companies based on different startup life-cycle stages. We followed the classification proposed by Klotins et al. [1]. Authors categorize startups into four stages e.g. inception, stabilization, growth, and maturity. Our studied startups belong to the last three stages i.e. stabilization, growth, and maturity.

We collected data primarily through semi-structured interviews, however, we considered additional public information about the companies as well. We conducted interviews remotely through Zoom and MS TEAMS. Each interview lasted from 60 to 90 min. The first author led the interviews together with the second author. Throughout the interviews, we followed a pre-designed interview protocol in which we defined open-ended questions regarding the introduction of the startup, product, team, measurement activities, and analytics practices undertaken. Afterward, we transcribed the interviews and analyzed the text data using open coding technique [2]. The coding process has resulted in renaming a few of the codes and adding more codes as the data analysis evolved. We carried out the data analysis process using Nvivo³, a qualitative research software. According to the agreement with the respondents in the informed consent form, we report our findings anonymously.

Case (A) is a startup company established in 2020 and delivers software-intensive services in the marketplace industry. It has already found its product-market fit and has a fair number of paying customers. Thus, we consider it in the stabilization phase. The team comprises six members, including the CEO and CTO of the startup. We interviewed the CEO of the startup.

Case (B) is a software startup that produces both B2B and B2C solutions in the business domain of sales and marketing. The startup was bootstrapped in the year 2018 and since then it has acquired several thousand paying customers.

¹ <https://www.linkedin.com/>.

² <https://www.crunchbase.com>.

³ <https://lumivero.com/products/nvivo/>.

Table 1. Case Overview

Case ID	Founded	Founder(s)	Team Size	Business Domain	Startup Stage
A	2020	1	6	Marketplace	Stabilization
B	2018	3	3	Sales & Marketing	Growth
C	2012	3	7	Transportation	Maturity

The company consists of a three-member team, headed by the CEO. Our data analysis suggests that the startup is in the growth phase as the efforts of the company are primarily shifted toward marketing and sales directions. We interviewed the CEO of the startup to understand their analytics practices (Table 1).

Case (C) is a unicorn startup providing transportation services through its software-intensive platform. The startup runs its analytics operations using its seven-member team and it has already acquired a very large customer base since its inception. It is currently focused on optimizing its operations and lies in the maturity phase according to the startup and product evolution stages. We interviewed the User Experience (UX) manager of the startup.

Lastly, it should be noted that the team size is not the same as all employees of the company. For the first two cases i.e. Case A and B, team size also refers to the overall employees of the company. However, for the third case, which is a unicorn startup, comprising hundreds of employees, we only considered the team mainly responsible for conducting analytics-related operations.

4 Findings

We identified nine key practices and classified them into four categories. We followed the analytics framework provided by Kayser et al. [19] to guide our classification of practices. This analytics process is structured in an attempt to succeed with analytics while moving from data to value. These high-level categories are business need, data collection, data exploration, and data communication. On the other hand, the identified individual key practices include focusing on a few KPIs, use of tools, communicating with the team, hiring people with an analytics background, blending data and intuition, understanding goals, collecting data early, sharing customer insights, and reaching out to customers. Figure 1 depicts an overall view of the identified analytics practices according to the analytics framework.

P1: Focusing on a Few Key Performance Indicators- We noticed that focusing on a few Key Performance Indicators (KPIs) is a principally visible key analytics practice in our data. Our data analysis also reveals that the tracked KPIs are associated with the goals that startups desire to achieve through analytics. In addition, the domain in which startups work also plays a significant role in deciding which KPI is important to monitor. For instance, the respondent from CASE C states: “*we understand the NPS and we have CSAT that we*

ID	ANALYTICS PRACTICE	BUSINESS NEED	DATA COLLECTION	DATA EXPLORATION	DATA COMMUNICATION
P1	Focus on a Few KPIs	✓	✓		
P2	Use of Tools		✓	✓	✓
P3	Communication with Team				✓
P4	Hire People with Analytics Background	✓	✓	✓	✓
P5	Blend Data and Intuition			✓	
P6	Understand Goals	✓			
P7	Collect Data Early		✓		
P8	Share Customer Insights				✓
P9	Reach out to Customers	✓	✓		

Fig. 1. Key Analytics Practices Mapped with the Analytics Framework, based on [19]

are monitoring the user in different moments of the journey to understand the CSAT”. Here, NPS refers to net promoter score while CSAT stands for customer satisfaction score, two key KPIs to measure customer experience. It is pertinent to state that startup C had a goal to improve customer satisfaction. Similarly, the founder of startup A stated that the number of customer applications is important to them. The founder indicated this as his startup is a marketplace and he thinks it is important to measure the network effect in this domain. The respondent continues expressing the need to collect the data that matters the most. He expressed it in the following words: “*Actually capture data that matters...that matters most*”. “*We are getting better and better*”, he further added.

P2: Use of Tools- All respondents from the studied startups reported the use of tools to support analytics setup in their companies regarding data collection, exploration, and communication. However, according to the interview data, startup A is only using Google Analytics while startup B is using Google Analytics as well as Facebook Analytics. Overall, respondents from both startups expressed that they are satisfied with the use of these tools and that these tools are satisfactory to achieve their analytical goals. A key determinant in selecting these tools is the “freemium feature”. On the other hand, as one might expect, startup C is using a number of tools to collect and explore data. It is worth noting that as the startup is mature, therefore, it heavily relies on internal tools to support the process. When it comes to collecting qualitative data, the startup is found using third-party tools like Qualtrics and Tableau.

P3: Communication with Team- Our data analysis reveals another common practice applied by startups in terms of communicating the data and insights with the team. All the respondents reported that they share and talk about data and interesting insights with other team members. For startup C, the com-

munication happens in a more formal way i.e. communicating insights through documents and using the language of data while other startups do it in an ad-hoc manner. The founder of startup A expressed this in the following words: *"We do talk to each other, not like groups, focus groups, or things like that... But we talk about analytics and interesting Data"*.

P4: Hire People with Analytics Background- All studied startups believe in hiring trained resources for collecting data and turning it into insights. Only startup B has not hired someone with skills, however, the founder expressed his desire to do so to utilize the full potential of analytics. It is expressed as: *"There is so much interesting data you will get from this... for our company, because there is no person, like really knowledgeable about this analytics... we don't really implement it like 100% but just 10%, and with just 10% we get like I think many insights"*. The CEO continued: *"If there's a person that knows this thing, I think they should do this"*.

Startup B has hired a resource to collect data, visualize data, and communicate it to the team. Similarly, startup C has three small yet dedicated sub-teams to manage analytics.

P5: Blend Data and Intuition- The respondents claimed that they use both data and intuition to act and make decisions. While one can expect it from startup C, however, interestingly, CEOs of other studied startups also highlighted this practice. For instance, the CEO of startup A alluded to his practice: *"maybe we should be a little more rational ...capture data. But if we care about data, we wouldn't start a company because...95% fails"*. The CEO of startup B holds the same belief: *"(decisions) based on this analytics only...No...I'm, I never like this"*.

P6: Understanding Goals- Our data analysis shows that it is a common practice to first understand the goals of the analytics. We also find that goal identification has a relation with defining KPIs to track. Therefore, in all cases, the informants explicitly reported the use of this practice. However, the goals varied in all our studied cases. For example, for startup A, user research and increasing network effect remain key goals. Startup B is involved in user research as well as having an increase in customer acquisition. On the contrary, startup C has a lot of goals e.g. improving efficiency, sustainability, testing hypotheses, and conducting A/B tests. The startup reported it: *"we have the goal to achieve it as a startup and to be self-sustainable as a startup"*. Interestingly, the startup also indicated their prior analytics goals and we find that those goals are now acting as primary goals for our other studied startups at earlier stages in comparison. It is apparent from these words: *"Two years ago, we had the goal of having more users, but now the goal we have, we can keep growing, but not investing a lot in new users, but most trying to be more efficient"*. This shows the evolution in terms of establishing analytical goals as the startup grows.

P7: Collect Data Early- The findings also show collecting data early is another key practice. Respondents from startups A and C explicitly indicated this. The respondent from startup A advised in the following words: “*Capture data early on so you understand*”. He also reflected on his mistake: “*the mistake we made is we implemented. We implemented Capturing the Data that matters most very late*”. The respondent from startup C also alluded to the same opinion of relying on (already collected) internal data when they are not so sure about solutions.

P8: Share Customer Insights- We observed that both startups A and B are implementing this analytics practice. The practice refers to promoting insights and numbers to public forums and using them for publicity and marketing purposes. For instance, the respondent from startup A articulated: “*Now we have data and we say...to clients, these are the data for you... And it helps us*”. A similar practice was stated by startup B. The respondent expressed: “*I just posted it like this*”. The core purpose was to “*to build the trust of the people like, oh, this application is working... many people [are] using it or something like that*”, the startup added.

P9: Reach Out to Customers- All the studied startups are found reaching out to customers to understand what users expect, discuss new proposals, take feedback, collect opinions, or add/remove product features. This is rather surprising because we expected mature startups to understand the qualitative data, however, startups at earlier stages have also been found practicing it. For instance, the CEO of startup A indicated: “*Customers’ input is very important and we do reach out to our customers and talk to them*”. He added further: “*Still we want to hear from the clients and the customers what they perceive as value*”. The same practice has been echoed by startup B in the following excerpt: “*they can request their likes, discuss with my team and then with the other members, and then if we have like... let’s say a conclusion like all this*”.

5 Discussion

Our research question was focused on determining how software startups are applying analytics. In this regard, we tried to identify the analytics practices. Our results depict nine analytics practices that are employed by software startups while applying analytics. These nine practices are classified into four categories in accordance with the analytics framework provided by Kayser et al. [19].

Contrary to our expectations, the study results have presented a handful of analytics practices that startups use. It might be related to the fact that we confirmed the use of analytics from our potential startup cases before proceeding with the data collection. Correspondingly, most of the practices are associated with the data collection phase, followed by business needs and data communication. This also accords with the findings of an earlier study where it is ascertained

that most analytics challenges occur within the data collection phase [8]. On the other hand, the data exploration phase contains the least number of reported analytics practices. Similarly, our results also support previous research (e.g. [4,5]) where it is indicated that analytics should be based on tools, techniques, and human sense-making.

While we report only on practices applied by startups at various stages of evolution, the frequency and intensity of implementing these practices vary from case to case. As one might expect, the startup at a mature stage is found applying practices intensively. The respondents from other startup companies have expressed their desire to strengthen analytics practices while highlighting the scarcity of resources. Consequently, their analytics process is somewhat ad-hoc in its current form. In addition, it is interesting to relate that, according to our data, startups at earlier stages learned from their mistakes. For example, collecting data early and hiring people with analytics backgrounds are example practices that have been adopted by startups after making mistakes. Likewise, understanding goals and focusing on a few KPIs are practices that have been highly recommended by the practitioner-oriented literature [17]. Our study empirically confirms the use of these practices by software startups.

Regarding threats to validity, our study has a number of limitations. For instance, we studied three software startups and interviewed one respondent from each case. Nevertheless, finding software startups with existent analytics set up and covering different startup stages are hard to manage. In the same way, although, we came across a number of other analytics practices reported by respondents in their startups, however, we did not report them in the findings as we had a limited number of agreements among the respondents.

6 Next Steps

Our findings enhance the scientific knowledge in disseminating analytics practices carried out inside software startups. However, considerably more work is needed to identify the analytics practices that are suitable for each stage of a software startup. Therefore, future research might put efforts into exploring key analytics practices that would be a good fit for a startup at a particular startup stage. In the future, we plan including more cases for each startup stage as well as interviewing multiple respondents from each case.

References

1. Klotins, E., et al.: A progression model of software engineering goals, challenges, and practices in start-ups. *IEEE Trans. Softw. Eng.* **47**, 498–521 (2021)
2. Saldaña, J.: *The Coding Manual for Qualitative Researchers*. Sage, Thousand Oaks (2021)
3. Yin, R.: *Case Study Research: Design and Methods*. Sage, Thousand Oaks (2009)
4. Bianchini, M., Michalkova, V.: Data analytics in SMEs: trends and policies. In: *SME and Entrepreneurship* (2019)

5. Sharma, R., Mithas, S., Kankanhalli, A.: Transforming decision-making processes: a research agenda for understanding the impact of business analytics on organisations. *Eur. J. Inf. Syst.* **23**, 433–441 (2014)
6. Begel, A., Zimmermann, T.: Analyze this! 145 questions for data scientists in software engineering. In: *Proceedings Of The 36th International Conference On Software Engineering*, pp. 12–23 (2014)
7. Sedkaoui, S.: How data analytics is changing entrepreneurial opportunities? *Int. J. Innov. Sci.* **10**(2), 274–294 (2018)
8. Rafiq, U., Melegati, J., Khanna, D., Guerra, E., Wang, X.: Analytics mistakes that derail software startups. In: *Proceedings of the 25th International Conference on Evaluation and Assessment in Software Engineering*, pp. 60–69 (2021)
9. Rafiq, U.: Towards understanding analytics in software startups. In: *2022 IEEE/ACM International Workshop on Software-Intensive Business (IWSiB)*, pp. 31–38 (2022)
10. Chitkara, B., Mahmood, S.M.J.: Importance of web analytics for the success of a startup business. In: *Batra, U., Roy, N.R., Panda, B. (eds.) REDSET 2019. CCIS*, vol. 1230, pp. 366–380. Springer, Singapore (2020). https://doi.org/10.1007/978-981-15-5830-6_31
11. Berg, V., Birkeland, J., Pappas, I.O., Jaccheri, L.: The role of data analytics in startup companies: exploring challenges and barriers. In: *Al-Sharhan, S.A., et al. (eds.) I3E 2018. LNCS*, vol. 11195, pp. 205–216. Springer, Cham (2018). https://doi.org/10.1007/978-3-030-02131-3_19
12. Chen, H., Chiang, R.H., Storey, V.C.: Business intelligence and analytics: from big data to big impact. *MIS Q.* 1165–1188 (2012)
13. Unterkalmsteiner, M., et al.: Software startups—a research agenda. *E-Informatica Softw. Eng. J.* **10**, 89–123 (2016)
14. Sutton, S.: The role of process in software start-up. *IEEE Softw.* **17**, 33–39 (2000)
15. Biesialska, K., Franch, X., Muntés-Mulero, V.: Big data analytics in agile software development: a systematic mapping study. *Inf. Softw. Technol.* **132**, 106448 (2021)
16. Davenport, T.H., Harris, J.G., Morison, R.: *Analytics at Work: Smarter Decisions, Better Results*. Harvard Business Press, Boston (2010)
17. Croll, A., Yoskovitz, B.: *Lean Analytics: Use Data to Build a Better Startup Faster*. O’Reilly Media, Inc., Newton (2013)
18. Duan, Y., Cao, G., Edwards, J.S.: Understanding the impact of business analytics on innovation. *Eur. J. Oper. Res.* **281**(3), 673–686 (2020)
19. Kayser, V., Nehrke, B., Zubovic, D.: Data science as an innovation challenge: from big data to value proposition. *Technol. Innov. Manag. Rev.* **8**(3), 16–26 (2018)

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.





Towards a X-as-a-Service Application in Industrial Laundry – A Case Study of Information Requirement Engineering in Emerging Data Ecosystems

Maximilian Werling^(✉), Alexandra Keller, and Heiner Lasi

Ferdinand Steinbeis Institute Heilbronn, Bildungscampus 9, 74076 Heilbronn, Germany
maximilian.werling@ferdinand-steinbeis-institut.de
<https://www.ferdinand-steinbeis-institut.de>

Abstract. In past years, Everything- or X-as-a-Service applications have left their mark on the consumer market, changing the landscape of on-demand delivery of goods and services. In recent years, these applications have also gained traction in an industrial context. However, companies are often unable to provide such a service on their own because they lack the necessary data or capabilities. To address this, companies are coming together to be part of larger Data Ecosystems. Based on a case study in industrial laundry, this paper explores the first steps towards a *Laundry-Finish-as-a-Service* application, focusing on the Information Requirement Concept. Based on Action Design Research as a guiding research process, three small to medium sized enterprises – a manufacturer of textile finishing machines, a manufacturer of gas burners as well as an industrial laundry – were scientifically accompanied over the course of eight months, with two workshops and several bilateral in-depth sessions. First results are presented and discussed in order to derive design principles for the Information Requirement Concept for a XaaS application in emerging data ecosystems.

Keywords: X-as-a-Service · Data Ecosystems · Information Requirement Concept

1 Introduction

As a result of the increasing penetration of internet technology in products and processes, a flexibilization of offerings and associated billing models – often under the label *Everything-as-a-Service* or *X-as-a-Service* (XaaS) – can be observed in recent years [1]. While the developments were initially applied in the consumer market in particular; for example, in the on-demand provision of films, music or other private consumer goods. XaaS applications are increasingly finding their way into an industrial context [2].

The core of XaaS solutions is always the on-demand or usage-based provision of (digital) service [3] – for example, the on-demand provision of compressed air [4]. Due to the higher requirements and greater complexity compared with conventional business models, individual companies are often not able to implement XaaS applications on their own [5]. Instead, several organizations align their capabilities and resources around a collaborative value proposition (i.e., for XaaS, the on-demand provisioning and billing of (digital) service) and share their data in data ecosystems [6–9]. The cross-company availability and processing of data and information are a central challenge in the realization of XaaS applications [10]. In the scientific literature as well as in practice, it can be observed that companies already have difficulties in an early phase of data sharing, namely in the identification and description of the data and information required for the realization of the XaaS solution [11]. This paper therefore addresses the following research question (RQ):

RQ: *How can different actors gather the required data and information as a basis for a XaaS application in emerging Data Ecosystems?*

To address the RQ, relevant background concepts are introduced. Then, the methodological approach and the case study on which this paper is based are described. Finally, the results are presented and discussed.

2 Background

2.1 X-as-a-Service in Data Ecosystems

Recent years have seen a surge of discussion in the academic literature on the topic of Digital Business Ecosystems (DBE) [7, 12]. Underlying this is an understanding of Business Ecosystems in which different actors come together in both collaborative and competitive ways to jointly produce new products, value propositions – ultimately new innovation – through combinations of the actors' different capabilities [8]. Adner (2017) highlights the importance of the collaborative value proposition, seeing it as the central aspect in the ecosystem to which actors align their activities and capabilities to contribute to the collaborative value proposition [6].

Data Ecosystems are understood as a special form of DBE and focus on the exchange of data between the actors of the ecosystem [9]. The exchange of data helps the actors to gain a broader database and can be the basis for the realization of new value propositions. Data Ecosystems are therefore understood in this paper as follows: they refer to multiple economic actors that come together in a collaborative as well as competitive manner and share data with each other with the goal of realizing a common value proposition [6, 9]. Since the case study described below involves three companies, we will refer to the environment as an emerging data ecosystem.

The definition of XaaS has changed over time, as it has been closely tied to technological developments in software architecture and cloud computing for many years [1]. In recent literature, the term is increasingly appearing in discussions of data-driven business models in ecosystem scenarios [3]. XaaS is the on-demand delivery of IT-based services, such as production capacity, that can be used as the basis for new business

models. Building on the previous sections, this paper understands XaaS as a collaborative value proposition to deliver IT-enabled, on-demand services. This understanding is further applied to the concept of *Laundry-Finish-as-a-Service*.

2.2 Information Requirement Concepts

The transfer of data between various stakeholders within a data ecosystem is crucial for the implementation of innovative digital business models [11]. However, both practical experience and scientific literature reveal a number of concerns about data sharing [10]. One challenge is that players often find it difficult to formulate their data and information needs transparently. Other players in data ecosystems also struggle to align their data and information offerings with existing needs. To describe the exchange of data and information in data ecosystems [13], information requirement concepts can be utilized. Information requirement concepts provide tools and criteria to describe data and information for exchange. They form the foundation for effective communication between data users and providers [14].

3 Methodology

3.1 Action Design Research

To address the research question presented in this paper, we utilized Action Design Research (ADR) methodology [15]. This approach involves the scientific investigation of problems while simultaneously developing practical solutions. The researchers were not merely observers of the processes but were actively involved in developing solutions via an interactive approach. The research approach is well-suited for exploratory scenarios that necessitate close collaboration with actors from the field. ADR aims at iteratively developing a socio-technical “ensemble artifact” [15], which takes the form of an information requirement concept for an XaaS application in industrial laundry. The concept was developed in two workshops and several bilateral one-on-one meetings. The researchers prepared and moderated the workshops, and photo protocols were used to document them. Memory protocols were used to document the bilateral one-on-one meetings between the organizations. These protocols and documentation served as an empirical basis for a qualitative analysis according to Mayring [16]. Figure 1 summarizes the activities:

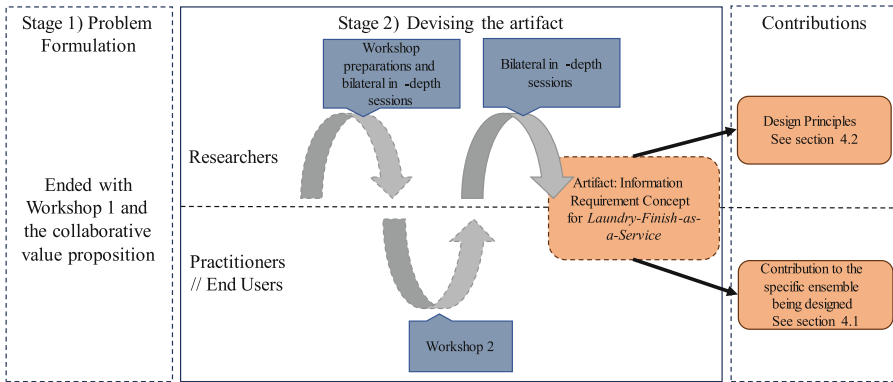


Fig. 1. Summary of the research process, based on Sein et al. [15]

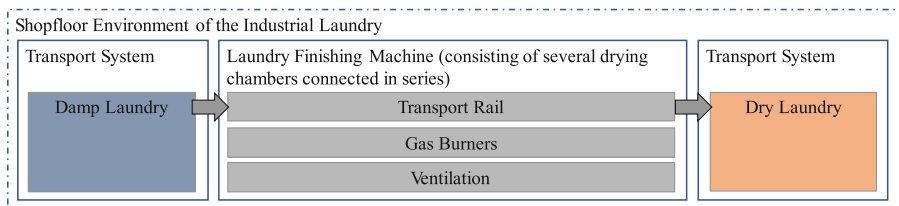
3.2 Case Study in Industrial Laundry

The case study (CS) and collaboration with business partners (BP) began in January 2022 and lasted approximately 8 months until August 2022. The project involved three companies, 1) a textile finishing machine manufacturer, 2) an industrial laundry, and 3) a gas burner manufacturer, to jointly conceptualize the collaborative value proposition: to come together in an emerging data ecosystem that enables the participating partners to develop a *Laundry-Finish-as-a-Service* application. The application intends to furnish textile finishing machines with data and information to execute laundry drying and finishing on demand based on the type, quantity, and condition of laundry available, as well as the efficient utilization of downtime. Table 1 summarizes the CS:

Textile finishing machines use gas burners to heat air and dry laundry gently and quickly. Damp laundry is loaded onto the machine via transport systems, guided through a drying chamber, and transferred back to the transport system as dry laundry. The drying chambers can also be ventilated to adjust the temperature and airflow in the chamber. Modern textile finishing machines have various drying programs to effectively dry laundry items of different fabrics. The responsibility of selecting the appropriate drying program lies with the operator of the industrial laundry. Inaccurate selection results in repeated feeding of laundry to the textile finishing machine, leading to wastage of time and energy. If the machine does not receive any laundry for an extended period, gas burners in the drying chamber can be adjusted to save energy. The project aims to enable the textile finishing machine to autonomously choose the suitable drying program and to optimize downtime by collecting pertinent contextual data, laying the groundwork for a *Laundry-Finish-as-a-Service* application. Considering the project partners, these machines are utilized in industrial laundry and are manufactured by the same company that designs the transport system and textile finishing machines. The drying chamber includes a gas burner designed and configured by the manufacturer of gas burners. Figure 2 schematically visualizes the process and involved objects of laundry finishing:

Table 1. Overview of the CS

Emerging Data Ecosystem		
Business Partner (BP)	Number of Employees	Participating Persons
(BP1) Textile Finishing Machine Manufacturer	~ 50	Managing Director, Head of Construction, Expert for Machine Control System
(BP2) Gas Burner Manufacturer	~ 10	Managing Director
(BP3) Industrial Laundry	~ 100	Managing Director, Expert for Machine Control Systems
Activities overview		
Workshop Date	Workshop Duration	Focus of Discussion
18.04.2022	~ 3 h	Conception of the collaborative value proposition. Elaboration of the benefits of the business partners
14.06.2022	~ 3 h	Specification of the relevant data and required information
Accompanying bilateral one-on-one meetings between the business partners and the research team		

**Fig. 2.** Schematic of the laundry finish process and objects involved

4 Results

4.1 Information Requirement Concept for *Laundry-Finish-as-a-Service*

The objective of the second workshop was to collect information on the requirements for implementing the *Laundry-Finish-as-a-Service* application. Representatives from all the companies involved met for this purpose. As part of the workshop preparation, we identified the relevant objects needed to realize demand-driven laundry finishing: the transport systems before and after the textile finishing machine, the machine itself and relevant sub-systems, such as the transport rail on which the laundry is guided through the drying chamber, the gas burner, and the ventilation. We then discussed which data and information from these objects would be required to realize the textile finishing on demand. The company representatives were able to determine directly whether the objects in question

can provide corresponding data and whether additional sensor technology should be added. Figure 3 summarizes the concept:

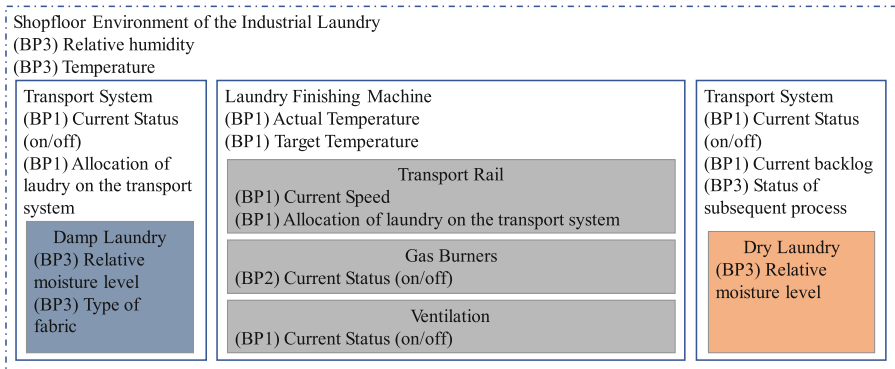


Fig. 3. Information Requirement Concept for a XaaS application Industrial Laundry

4.2 Emerging Design Principles

From the experiences of the activities of the CS, design principles (DP) can be extracted for the design of information requirement concepts in emerging data ecosystems:

Bringing together diverse stakeholders in companies was found to be effective. Although a comprehensive understanding of the finishing process was necessary, evaluations from domain experts on machine controls for various objects and machines quickly identified available or potentially obtainable data and information. While the managing directors contributed their domain-specific knowledge, such as the managing director of the industrial laundry outlining the specifics of drying different fabrics and the interplay between drying and surrounding processes, experts for design and machine controls from BP1 and BP3 also participated in the workshops.

DP1: Assemble the stakeholders possessing both domain-specific and technical expertise to identify relevant data and information for the intended service and to consider technical feasibility.

The collaborative discussion in the workshops between stakeholders provided a secondary benefit. During the initial workshop, potential areas for optimization were identified, particularly through the discussions between the representatives of BP2 and BP3, who would otherwise not be in direct contact with each other. The comprehension of the interaction between individual objects and components during real-world use was a crucial requirement for conceiving the *Laundry-Finish-as-a-Service* application as it revealed a divergence between the intended and actual use of the laundry finishing machine. Achieving this level of transparency ensures that requirements are established based on real-life scenarios.

DP2: Bring together stakeholders responsible for relevant processes helps to establish transparency regarding the use and interdependence of relevant objects. This also ensures that requirements are derived from real-life scenarios.

Particularly in an environment dominated by small and medium-sized enterprises (SMEs), where familiarity with digital service design may not be a prerequisite, focusing on objects proved effective in identifying information needs. The object-focused approach was intuitive for even those without IT-specific expertise, facilitating goal-oriented discussions. Building the discussion around objects, like the transportation system or the textile finishing machine and its components, that were familiar to all participants was found to be helpful, especially in the second workshop and in the more in-depth individual discussions.

DP3: Focus on objects to provide an intuitive method for participants with no or little prior experience in digital service creation for the assessment of the information requirements.

In-depth discussions with machine control experts revealed that much data and information exists solely within the control systems of respective objects. However, to implement the concept, data must be aggregated and analyzed across different objects and companies. Acquiring relevant data from machine control systems and providing adequate connectivity to bridge different systems, would require considerable effort, the experts said. Utilizing Internet-of-Things technology can achieve interoperability between object data [17].

DP4: To aid interoperability use Internet-of-Things technology where possible to interconnect different smart objects.

5 Conclusion and Limitations

This paper outlines a concept for object-focused information requirements engineering in emerging data ecosystems. The concept serves as the basis for implementing a *Laundry-Finish-as-a-Service* application. This application delivers dry laundry on-demand, based on the type, quantity, and condition of laundry in an industrial laundry context. To meet the information needs of various companies' objects, they are collaborating to share data in an emerging data ecosystem. This effort was the first foray into digital service creation for the participating companies, stemming from the textile finishing machine manufacturer's desire to innovate its service model. For practitioners, the paper provides insight into the first steps of the process of developing an XaaS application. DP based on the experiences of the described activates prescribe actions for conceptualizing information requirements within an emerging data ecosystem, specifically targeting SMEs [18]. From a scientific perspective the study adds to body of literature outlining activities in digital service innovation in the context of SMEs [5, 19].

Unfortunately, the cooperation and advancement of the *Laundry-Finish-as-a-Service* application was suspended in early September 2022 owing to the European energy crisis that arose at that time. With a focus on new challenges in the operational business, the business partners could no longer guarantee the necessary commitment to the further development of the XaaS application. Thus, the research team intends to continue the collaboration and specifically implement a prototype of the conceptualized application. The work leaves ample room for further development and research. From a scientific standpoint, numerous intriguing questions will emerge throughout the project's progression, such as the critical elements in crafting appropriate billing models or inquiries

concerning liability and warranties for XaaS applications. The authors aim to examine these matters in future research.

Finally, this study has limitations. The generalizability based on a single case study must be considered. The study presents the methodological approach and results of information requirement engineering in an emerging data ecosystem within the context of industrial laundry. It is essential to note that not all findings in this study can be applied to the development of XaaS applications in general. However, the authors argue that they have achieved an apt abstraction of the outcomes with the DP discussed, enabling their portability to diverse contexts [18].

References

1. Duan, Y., Fu, G., Zhou, N., Sun, X., Narendra, N.C., Hu, B.: Everything as a service (XaaS) on the cloud: origins, current and future trends. In: 2015 IEEE 8th International Conference on Cloud Computing (CLOUD), IEEE Computer Society, Editor, New York City, NY, USA, June 27 - July 02 2015
2. Guo, J., Nikolay, M., Wan, G.: the partner ecosystem evolution from on-premises software to cloud services: a case study of SAP. In: Proceedings of the 23rd Pacific Asia Conference on Information Systems (PACIS), Association for Information Systems (AIS), Editor, PACIS 2019, X'ian, China, 08–12 July (2019)
3. Riasanow, T., Krcmar, H.: Everything as a Service (XaaS). In: Kollmann, T. (ed.) *Handbuch digitale Wirtschaft*, pp. 985–996. Springer, Wiesbaden (2020). https://doi.org/10.1007/978-3-658-17291-6_70
4. <https://www.kaeser.com/int-en/company/about-us/industrie-4-0-iiot>. Accessed 19 Apr 2023
5. Weber, P., Morar, D., Lasi, H.: Transforming value chains into internet-based ecosystems: a testbed approach. In: Kocaoglu, D.F. (ed.) *Proceedings of the PICMET 2018 Conference, PICMET 2018, Honolulu, Hawaii, USA, August 19–23 2018*
6. Adner, R.: Ecosystem as structure: an actionable construct for strategy. *J. Manag.* **43**(1), 39–58 (2017)
7. Baumann, S., Leerhoff, M.: Networks, platforms, and digital business ecosystems: mapping the development of a field. In: Baumann, S. (ed.) *Handbook on digital Business Ecosystems: Strategies, Platforms, Technologies, Governance and Societal Challenges*. Edward Elgar Publishing, Cheltenham (2022)
8. Moore, J.F.: Predators and prey: a new ecology of competition. *Harv. Bus. Rev.* **71**(3), 75–86 (1993)
9. Oliveira, M.I.S., Lóscio, B.F.: What is a data ecosystem? In: *Proceedings of the 19th Annual International Conference on Digital Government Research: Governance in the Data Age, Association for Computing Machinery (ACM), Delft, Netherlands, May 30–June 01 2018*
10. Gelhaar, J., Otto, B.: Challenges in the emergence of data ecosystems. In: *Proceedings of the 24th Pacific Asia Conference on Information Systems (PACIS), Association for Information Systems (AIS), PACIS 2020, Dubai, UAE, June 22–24 2020*
11. Azkan, C., et al.: Anreizsysteme und Ökonomie des Data Sharings: Handlungsfelder des unternehmensübergreifenden Datenaustausches und Status quo der deutschen Wirtschaft (2022)
12. Tsujimoto, M., Kajikawa, Y., Tomita, J., Matsumoto, Y.: A review of the ecosystem concept: towards coherent ecosystem design. *Technol. Forecast. Soc. Chang.* **136**, 49–58 (2018)
13. Susha, I., Janssen, M., Verhulst, S.: Data collaboratives as a new frontier of cross-sector partnerships in the age of open data: taxonomy development. In: *Proceedings of the 50th Hawaii International Conference on System Sciences, Association for Information Systems (AIS), HICSS 2017, Hilton Waikoloa Village, Hawaii, January 04–07 2017*

14. Lachenmaier, J.F., Lasi, H., Kemper, H.-G.: Entwicklung und Evaluation eines Informationsversorgungskonzepts für die Prozess- und Produktionsplanung im Kontext von Industrie 4.0. In: *Wirtschaftsinformatik 2015 Proceedings, Association for Information Systems (AIS), WI2015*, Osnabrück, Germany, March 04–06 2015
15. Sein, M.K., Henfridsson, O., Purao, S., Rossi, M., Lindergen, R.: Action design research. *MIS Q.* **35**(1), 37 (2011). <https://doi.org/10.2307/23043488>
16. Mayring, P.: Qualitative Content Analysis: Theoretical Background and Procedures. In: Bikner-Ahsbals, A., Knipping, C., Presmeg, N. (eds) *Approaches to Qualitative Research in Mathematics Education. Advances in Mathematics Education*, pp. 365–380. Springer, Dordrecht (2015). https://doi.org/10.1007/978-94-017-9181-6_13
17. Noura, M., Atiquzzaman, M., Gaedke, M.: Interoperability in internet of things: taxonomies and open challenges. *Mob. Netw. Appl.* **24**(3), 796–809 (2019)
18. Chandra, L., Seidel, S., Gregor, S.: Prescriptive knowledge in IS research: conceptualizing design principles in terms of materiality, action, and boundary conditions. In: *Proceedings of the 48th Hawaii International Conference on System Sciences, IEEE Computer Society, HICSS 2015*, Kauai, Hawaii, January 05–08 2015
19. Kurrle, S., Hiller, S., Weber, P., Lasi, H.: Implementing a proof-of-concept in IoT-ecosystems: a case study in the hospitality industry. In: Kocaoglu, D.F. (ed.) *Proceedings of the PICMET 2022 Conference, PICMET 2022*, Portland, Oregon, USA, August 07–11 2022

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.





Software Startup Ecosystem in Namibia

Hilma Aludhilu¹(✉)  and Erkki Sutinen² 

¹ University of Eastern Finland, Joensuu, Finland
hilmaalu@uef.fi

² University of Turku, Turku, Finland
erkki.sutinen@utu.fi

Abstract. The number of software startups in Namibia has increased over the last decade, although most of them do not survive for long in the industry. For software startups to thrive, a suitable ecosystem is required to support them as the sustainability of startups is determined by the actions and interactions of the ecosystem actors. We aimed to gain a better understanding of the current software startup ecosystem in Namibia, emphasizing how the startup is connected to and supported by other actors in the ecosystem. Understanding the ecosystem will assist in informing future support needed by software startups to increase their sustainability and the growth of the ecosystem. An online questionnaire was employed to collect data from participants from software startups, as well as institutions that support software startups and entrepreneurs in Namibia. The results show that the Namibian software startup ecosystem is still in its early development stages and offers limited assistance for startups to grow. Access to finance is a challenge for startups, as most of the startups are founded and supported by personal funds, and few are funded by investors and Venture Capital funds and receive little to no financial support from the government. The universities play a role in supporting software startups through software development and entrepreneurial education, and training. Incubators and accelerators, although not a lot in the ecosystem, offer software entrepreneurs mentorship and a supportive environment to grow their businesses. The startups require more funding, access to resources, mentorship, and networking opportunities from other ecosystem actors.

Keywords: Software Startups · Startup Ecosystem · Software Entrepreneurship

1 Introduction

Software startups are new software venture companies founded by individuals who are attempting to break into the market with limited funds and experience. Nanthaamornphong and Wetprasit [1] defined a software startup as a business with the potential to expand its operations by exploring new business models and leveraging technology to conduct business based on innovative ideas. Software startups are important producers of innovation, products, and services as they focus on creating and launching innovative software solutions within a limited time and with minimal resources [1–5]. Although fewer than anticipated, Namibia saw an increase in the number of software startups

over the last decade (2013 to 2022). In Namibia, software startups are established to satisfy the growing need for software solutions [6, 7]. In addition to providing online services to Namibians, software startups aim to develop technology solutions for the health, agricultural, educational, and economic sectors.

Startups play a pivotal role in enhancing the socioeconomic progress of society [8] as they boost employment opportunities through the creation of new jobs. Software startups can drive job creation in Namibia and positively impact overall economic productivity. However, most startups do not survive for long in the industry due to several challenges such as the lack of resources and producing products that do not stay long in the market. For software startups to be supported and succeed, suitable ecosystems are required to assist them to thrive in the industry. A software startup ecosystem, which includes a variety of agents, is necessary and essential for the growth of startups [9, 10]. The software startup ecosystem in Namibia consists of organizations, institutions, and policies that contribute to the creation, growth, and sustainability of software startups in the country. The sustainability of startup companies is determined by the actions and interactions of the ecosystem elements [11]. It is important to study software startups and their ecosystems to assist individuals aspiring to create software startups and those seeking to provide support to the startups [1]. This research, therefore, aims to study the Namibian software startup ecosystem, as limited studies have focused on understanding the current situation within Namibia's software startup ecosystem. The study is unique in its focus on factors that are specific to Namibia's software startup ecosystem, considering the country's economic and social context. By identifying the challenges and opportunities facing software startups in Namibia, this study provides new insights into how the ecosystem can be supported.

The study aims to answer the following research question: What is the current state of the software startup ecosystem in Namibia? We collected data using a questionnaire from participants from startups, and from institutions that support software startups. We described the components present in the Namibian software startup ecosystem and conducted a SWOT analysis of the startup ecosystem. The remaining sections of this paper are as follows: Sect. 2 presents the related work and Sect. 3 presents the research design. Section 4 presents the results, which are then discussed in Sect. 5. Section 6 concludes the paper.

2 Related Work

There has been a notable research emphasis on software startup ecosystems due to its significant economic and societal impact. Numerous studies have examined the characteristics, challenges, and opportunities of software startup ecosystems around the world. Kon et al. [12] studied the Israeli software startup ecosystem, known for being highly productive, and developed a framework based on their research. The study highlights the ecosystem's strengths, including high-quality software engineering talent and strong government support, and identified limited access to venture capital and a shortage of management skills as challenges. Asmoro and Nugroho [13] developed a conceptual framework for the software startup ecosystem in Indonesia based on a literature review and expert interviews. The conceptual framework established connections between various aspects of startups such as their founders, sociocultural, institutional, technological,

methodological, and educational factors, as well as their environment and ecosystem. The framework is designed to provide a comprehensive understanding of these factors and their influence on the growth and success of startups. Nanthaamornphong and Wetprasit [1] analyzed Thailand's software startup ecosystem, exploring its current state. The study identifies the ecosystem's key players and examines government policies and initiatives to support the ecosystem, such as funding programs, tax incentives, and the establishment of startup incubators and accelerators. Despite the ecosystem's potential to become a regional hub for innovation and entrepreneurship, the authors acknowledge challenges, such as limited access to capital. The study concludes by offering recommendations for stakeholders, especially the government to support the ecosystem's growth and development.

Research on software startup ecosystems has yielded valuable insights into various aspects, such as the role of ecosystem components, government policies and support, and funding sources. Despite progress made in understanding software startup ecosystems, there are still many challenges and opportunities to explore in different contexts, especially in emerging economies like Namibia. The software startup ecosystem in Namibia remains largely unexplored, making it difficult for policymakers and investors to effectively support and advance the ecosystem, hindering its ability to contribute to job creation and economic development in the country. Therefore, conducting a comprehensive study of the software startup ecosystem in Namibia is crucial for identifying key challenges and opportunities, and developing strategies to address them.

3 Research Design

This study aims to address the research question, "What is the current state of the software startup ecosystem in Namibia?". To achieve this goal, we carried out an exploratory study about the startup ecosystem, using the questionnaire as the instrument to collect data. The questionnaire is divided into three parts. The first part focuses on the startup's inception, including the motivation and funding sources used. The second part examines the institutions that support software startups. The third part explores the strengths, weaknesses, opportunities, and threats of the Namibian software ecosystem, using the SWOT analysis framework inspired by Humphrey [14]. The current situation of the ecosystem is covered by the strengths and weaknesses while the opportunities for further development and threats of the Namibian software startup ecosystem for the future are covered by the opportunities and threats. The questionnaire was piloted with four participants to ensure clarity and relevance and the link (<https://shorturl.at/uMNPO>) was distributed through email and WhatsApp to six participants known to the researchers who own or work for software startups or work for institutions that support startups or from the software development community. Using the snowball sampling method, the selected respondents were advised to forward the link to other potential respondents, and a total of 15 responses were received. A total of 8 respondents are from startups, 3 are from organizations that support startups and 4 are from the software development community.

The analysis of data from the questionnaire, which had both close-ended and open-ended questions, followed a mixed-methods approach. For close-ended data, quantitative

analysis involved calculating frequencies and percentages to summarize participants' responses. For open-ended data, thematic analysis was employed by two researchers through a collaborative approach. One researcher conducted the initial coding process, while the other validated and reviewed the analysis. The analysis started with the familiarization data, followed by data exploration guided by the following statements: establishment of software startups, institutions that support software startups and entrepreneurs, and SWOT of the ecosystem. The researcher assigned initial codes to text and the codes are then refined and organized into broader categories that give rise to themes. Once the coding process was complete, the second researcher independently reviewed the generated codes and categorized themes. Their role was crucial in ensuring the validity and reliability of the analysis.

4 Results

The data collected were analyzed according to three categories: establishment of software startups, institutions that support software startups and entrepreneurs, and SWOT analysis of the Namibian software ecosystem.

4.1 Establishment of Software Startups

The results show that software startups are mostly created because of the existence of a good idea that fills a gap in the market (53%). Following this, some startups are founded due to unemployment (20%) and low earnings in the current job (20%). A smaller portion of startups are founded to make their products and services more competitive in the rapidly digitizing environment (7%). The software entrepreneurial attitude is influenced by both intrinsic, extrinsic, and contextual motivations. Intrinsic motivation is driven by internal factors such as the “*desire to solve local problems with software*”, “*the love for technology*”, “*interest in technology, especially software*” and the need for “*freedom to explore technologies*”. Extrinsic motivation is driven by external factors such as “*market needs*” with “*vast available opportunities*”, “*entrepreneurial education*”, and the influence of culture, media, and society. According to the results, “*unemployment*”, “*market demand and opportunities*” are contextual motivations that drive individuals to initiate software startup ventures. Regarding the sources of funding, startups mostly use personal savings or funds (93%) and funds from family, relatives, or friends (73%), and from innovation competitions (67%). They also use funds from crowdsourcing platforms (13%), loans from banks (13%), or funds from incubators (13%). Few (7%) get funds from project finance. When starting the business, software startups are mostly faced with challenges of lack of funding (93%), followed by dealing with legal, bureaucratic, or procedural issues (60%), target market approach (40%), finding a suitable operating space (33%), finding, and using appropriate technology (27%), finding staff (20%) and finding partners (13%). When it comes to technological aspects that support software start-ups, the respondents mentioned that Open-source software is widely used by software startups to develop their products. A respondent said that “*open-source software allows startups to make use of software at a limited cost.*” Respondents also indicated that startups make use of agile methods in their development and make use of digital payment processing systems.

4.2 Institutions that Support Software Startups and Entrepreneurs

The institutions that promote software startups and entrepreneurship include funding institutions, educational institutions, incubators, and accelerators. Funding institutions that support software entrepreneurship include angel investors such as The Namibia Business Angel Network (NABAN) and venture capital funds that invest in startups. The startups also receive funding through competitions funded by companies like Standard Bank, First National Bank, and Mobile Telecommunications Company (MTC). Educational institutions that promote software entrepreneurship are the Namibia University of Science and Technology (NUST) and the University of Namibia (UNAM). They provide education and training to software developers who become software entrepreneurs. A respondent indicated that “*Namibia University of Science and Technology has hackathons and sponsorship opportunities*”. Another respondent also said that “*NUST hosts events that developers from startups can attend, to improve when it comes to software engineering and entrepreneurship*”. NUST also has the High-Tech Transfer Plaza Select (HTTPS) state-of-the-art facility which aims to facilitate technologically accelerated innovations and offers a unique environment for startups. Incubators and accelerators such as The Namibia Business Innovation Institute (NBII), The Namibia Investment Promotion and Development Board (NIPDB), and StartUp Namibia promote software entrepreneurship in Namibia. A respondent indicated that “*NIPDB offers capacity building to TechNovators*”. The respondent indicated that the NBII is one of the co-working spaces where software startups meet. Other respondents indicated that “*NBII provides space for software startups to meet and take part in the training they offer*”, and “*mentors software entrepreneurs with innovative ideas to start their businesses*”. When it comes to the legal frameworks in Namibia that influence software startups, startups are influenced by labor law, BIPA regulations, and copyright protection of innovative ideas.

4.3 SWOT Analysis of Software Startups Ecosystem

When it comes to the current strengths of the software startup ecosystem, developers in the software startups are determined to work hard together for their startups to survive. A respondent indicated that “*Namibia is at its peak of digital transformation, and this is a major motivation for startups*”. Innovative ideas also exist, as one respondent indicated that “*there are a lot of creative ideas for software entrepreneurs*”. Another opportunity is also associated with “*localized solutions that can have a regional and international impact*”. The current weaknesses of the software startup ecosystem include a lack of funding and mentorship, a lack of knowledge on pricing, and the small market. The lack of funding is a major weakness as a respondent indicated that “*There is a lack of funding for software startups. More funding institutions could come on board to provide funding.*” Another respondent said that “*people have great ideas, however, a lot of them lack startup capital.*” Another weakness is that startups mostly only develop for the local market and hardly for the global market. The startups need to also think of getting their products out to the global market. A respondent indicated that “*The population in urban areas and internet connection is not enough to make software entrepreneurship a niche market.*” Also, most of the software developers in the startups do not have the

needed entrepreneurial skills, and business knowledge, which affects the operation of the startup.

The future opportunities in the software startup ecosystem include software development and entrepreneurial education of new software entrepreneurs, the emerging market as “*more people will know how to make use of software solutions in the future*” and innovative ideas among software developers. Incubators and accelerators can offer mentorship to new software entrepreneurs and a supportive environment in which to grow their businesses. A respondent wrote that “*the future is bright as software opportunities will always be there. With the economic crisis, youths will come up with even greater innovative solutions.*” The threats facing the ecosystem include competition and poor software quality. Software startups face the threat of competing with other established software companies in the country. A respondent indicated that “*More people like using international software that is already established.*” Also, the ecosystem faces threats when it comes to software quality as customers expect software of high quality which the startups may not be able to provide due to limited resources. The summary of the Namibian software startup ecosystem as a SWOT analysis is shown in Table 1.

Table 1. Software startup ecosystem summarized as a SWOT analysis.

Positive	Negative
Strengths <ul style="list-style-type: none"> • Collaboration • Determination to work hard • Peak of digital transformation • Existence of innovative ideas • Localized solution 	Weaknesses <ul style="list-style-type: none"> • Hard to access capital • Lack of funding • Limited entrepreneurial skills • Lack of governmental support • Small local market
Opportunities <ul style="list-style-type: none"> • Entrepreneurial education and mentorship for entrepreneurs • Software development skills and technology training • Emerging markets • Innovative solutions 	Threats <ul style="list-style-type: none"> • Competition • Poor software quality • Lack of funds

5 Discussion

The software startup ecosystem consists of startups, entrepreneurs, government and private investors, universities, and support organizations, which are essential to the sustainability of the startup. Funding is a crucial component for the operation of software startups. However, it is also a major challenge for startups, as only a few receive funding from private funding bodies and little to no financial support from the government. Startups in Namibia experience a constraint of restricted access to capital, which is a challenge also encountered by startups in other global contexts [1]. However, the severity

of this challenge appears to be amplified within the Namibian context, primarily due to the economic and funding dynamics in the country. This is a concern as startups' growth and viability are put at risk by the lack of funding and a significant reliance on personal capital [11].

Compared to other ecosystems such as that of Israel [12], Namibia's ecosystem has significantly fewer incubators, accelerators, co-working spaces, venture capital, and angel investor networks. Incubators and accelerators can assist in offering mentorship to software entrepreneurs and a supportive environment to grow their businesses. However, Namibia has few startup incubators and accelerators that can foster the growth of software startups. The lack of startup incubators and accelerators in Namibia requires a holistic strategy that incorporates efforts from educational, governmental, and private sectors, as well as the international community. Collaborations with educational institutions would enable knowledge-sharing, and research support, although Intellectual Property Rights (IPRs) can be a complex issue when it comes to collaborations with higher education institutions (HEIs). Government intervention could encompass introducing policies and incentives, particularly in the tax sector, that can encourage the establishment of incubators and accelerators. Engaging the private sector could increase the resources available to startups through financial investments and partnerships with incubators and accelerators. Additionally, forging international partnerships with established incubators and accelerators could enhance global expertise and networks in the ecosystem.

Namibia's local software market is relatively smaller and more nascent compared to the mature markets in established software ecosystems such as those of Israel and Thailand [1, 12]. The customer base in Namibia is limited due to the country's smaller population and economy. The small market size means that startups have a smaller customer base for their products and services to target in the country. Startups can maximize the potential within the local market through niche targeting, while also considering opportunities for growth beyond national borders. The challenge of a small market should encourage startups to think globally from the outset and to seek customers and market needs beyond Namibia's borders. This mindset can drive innovation, diversify income streams, and potentially lead to stronger international connections. Poor quality software poses a significant threat to the Namibian software ecosystem. This threat emanates from various factors, including limited resources and skill gaps. Poor quality software limits the growth of local software startups by tarnishing their reputation and hindering customer satisfaction and trust. Therefore, fostering a culture of quality assurance is crucial to mitigate this threat. Additionally, enhancing product quality is a key component for startups looking to expand beyond local boundaries and enter regional or international markets.

The identified opportunities in the ecosystem offer space for new directions and strategies to strengthen and expand the ecosystem. As previous research has highlighted the need for government support to foster ecosystem growth [1], we also call on the government to support Namibian startups. The government still has a lot to contribute to the ecosystem, especially the Ministry of Information and Communication Technology should foster innovation by establishing conducive environments through permissionless innovation policies instead of restrictive ones to permit or foster a culture of experimentation. Our recommendation to practitioners operating within the Namibian software

startup ecosystem is that they should concentrate on problem-solving to address customer needs and demands, attract investment, establish strong networks, utilize available resources, and adopt emerging digital technologies. By implementing these measures, practitioners can position themselves favorably for success within the constantly evolving industry. The study contributes to research by providing a case study of an emerging software startup ecosystem in a developing country, offering insights into the challenges and opportunities unique to the context.

The limitation of the study is the relatively small scale and emerging nature of the ecosystem itself. As Namibia is still developing its technology industry, the sample size of startups, entrepreneurs, and ecosystem stakeholders is limited, leading to challenges in obtaining comprehensive data and insights. The data is interpreted within the context of the ecosystem's current developmental stage. Although the study provides valuable insights into the current state of the ecosystem, the findings cannot be generalized. The results are however valuable for informing ecosystem improvement efforts, guiding investment decisions, and facilitating collaboration among stakeholders, all of which collectively support the ecosystem's growth and development within its specific context.

Future research can concentrate on several strategic directions, directed by the findings of a SWOT analysis of Namibia's software startup ecosystem carried out in this study. It could involve capitalizing on identified strengths through initiatives that enhance existing advantages and address the weaknesses by investigating innovative approaches to overcome challenges related to funding and local market size. Exploring opportunities pinpointed by the analysis could lead to research on innovative solutions and entrepreneurial skills amongst software developers. Similarly, mitigating threats could entail in-depth studies on improving software quality and competitive strategies. Future work could also focus on exploring how modern entrepreneurial approaches such as Lean Startup is adopted in the Namibian ecosystem.

6 Conclusion

The study shows that the software startup ecosystem is still developing and offers limited assistance for startups to grow. Software entrepreneurs in the ecosystem are highly influenced by the family, culture, media, innovative ideas, and society in which they live. Namibian universities, incubators, accelerators, and angel investors support the startups, although limited. Access to finance continues to be a challenge for startups, as we identified the recurring need for funding. The challenges and threats identified in the ecosystem need to be addressed to promote the growth and success of software startups, as startups have the potential to significantly contribute to Namibia's economy, especially through job creation.

References

1. Nanthaamornphong, A., Wetprasit, R.: Thailand's software startup ecosystem. In: Nguyen-Duc, A., Münch, J., Prikładnicki, R., Wang, X., Abrahamsson, P. (eds.) *Fundamentals of Software Startups*, pp. 195–213. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-35983-6_12

2. Sutton, S.M.: The role of process in a software start-up. *IEEE Softw.* **17**(4), 33–39 (2000)
3. Klotins, E.: Using the case survey method to explore engineering practices in software start-ups. In: 2017 IEEE/ACM 1st International Workshop on Software Engineering for Startups (SoftStart), pp. 24–26. IEEE (2017)
4. Klotins, E., Unterkalmsteiner, M., Gorschek, T.: Software-intensive product engineering in start-ups: a taxonomy. *IEEE Softw.* **35**(4), 44–52 (2018)
5. Berg, V., Birkeland, J., Nguyen-Duc, A., Pappas, I.O., Jaccheri, L.: Software startup engineering: a systematic mapping study. *J. Syst. Softw.* **144**, 255–274 (2018)
6. Nehemia-Maletzky, M., Iyamu, T.: Challenges of software developers: the Namibian government experience. *Int. J. Inf. Technol. Manage.* **17**(3), 237–255 (2018)
7. Nelulu, J., Mufeti, T.K.: An exploratory study of the development practices used by software entrepreneurs in Namibia. In: Halberstadt, J., Marx Gómez, J., Greyling, J., Mufeti, T.K., Faasch, H. (eds.) *Resilience, Entrepreneurship and ICT*. CSEG, pp. 79–93. Springer, Cham (2021). https://doi.org/10.1007/978-3-030-78941-1_4
8. Gonçalves, K.L.F., Gonçalves, R.F.: Nomenclatures, terminologies and classification of startups: a multivocal literature review. *Res. Soc. Dev.* **10**(4), 1–12 (2021)
9. Cukier, D., Kon, F.: A maturity model for software startup ecosystems. *J. Innov. Entrep.* **7**(1), 1–32 (2018)
10. Cukier, D., Kon, F., Gjini, E., Wang, X.: Startup ecosystem maturity and visualization: the cases of New York, Tel Aviv, and San Paolo. In: Nguyen-Duc, A., Münch, J., Prikladnicki, R., Wang, X., Abrahamsson, P. (eds.) *Fundamentals of Software Startups: Essential Engineering and Business Aspects*, pp. 179–194. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-35983-6_11
11. Ziakis, C., Vlachopoulou, M., Petridis, K.: Start-up ecosystem (StUpEco): a conceptual framework and empirical research. *J. Open Innov. Technol. Mark. Complex.* **8**(1), 35 (2022)
12. Kon, F., Cukier, D., Melo, C., Hazzan, O., Yuklea, H.: A panorama of the Israeli software startup ecosystem (2014)
13. Asmoro, A., Nugroho, L.E.: Software startup ecosystem in Indonesia: a conceptual framework. In 2018 4th International Conference on Science and Technology (2018)
14. Humphrey, A.S.: SWOT analysis. *Long Range Plan.* **30**(1), 46–52 (2005)

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.





Industry Expectations for Product Ops Professionals: A Review of Job Advertisements

Bogdan Moroz, Andrey Saltan^(✉), and Sami Hyrynsalmi

LUT University, Lahti, Finland

{bogdan.moroz, andrey.saltan, sami.hyrynsalmi}@lut.fi

Abstract. This paper aims to provide clarity on the emerging role of Product Operations (Product Ops) in software-developing organizations. As organizations increasingly acknowledge the benefits of having a dedicated Product Ops function, inconsistencies and uncertainties surrounding the role's responsibilities persist. By analyzing job advertisements for Product Ops positions, this study identifies and categorizes the expected skills and responsibilities, subsequently ranking them by frequency. The results offer a clearer understanding of the Product Ops role based on industry expectations expressed in the job postings.

Keywords: Product Ops · Product Operations · Software Production

1 Introduction

Product Operations (Product Ops) is an emerging function in software-developing organizations, designed to bolster software product management (SPM) by incorporating a distinct operational aspect. This function is defined as one that makes companies more efficient and allows them to scale without friction, and empowers product teams in four dimension [6]:

1. Ensuring that software product managers regularly receive clean and reliable data to base their decisions on (**data management dimension**);
2. Managing the tooling, processes, and infrastructure used by the product team, establishing and communicating best practices (**tool and process management dimension**);
3. Allowing software product managers to focus on core SPM work by reducing the administrative burden and acting as a pro-active assistant (**operational complement dimension**);
4. Fostering cross-departmental and cross-team communication, collaboration, and coordination, ensuring alignment and preventing silos (**collaboration dimension**).

Despite the growing recognition of the benefits associated with the emerging business function of Product Ops, there is no consensus on the role's specific

responsibilities and how the Product Ops role should be implemented [6, 7]. To address this, the paper evaluates job advertisements (job ads) for Product Ops positions to provide a coherent portrait of a Product Ops professional.

Rafaeli and Oliver [8] have argued that job ads are a relevant form of organizational communication and demonstrate the richness and complexity of information that can be shared via such ads. Most ads have a similar “skeleton”, consisting of four parts: the identity of the hiring organization, the human resources demands it needs to meet, the description of the requirements to fulfill these demands, and contact information for applicants. While this skeleton is sufficient to fill the employment need, most ads go further and include information regarding the legal requirements that the organization complies with, as well as the organization’s size, financial situation, history, expected future products, values, and culture [8]. The authors argue that job ads can also be aimed at the general public (including potential investors), current employees, and other organizations [8]. The objectivity of the ads must therefore be approached with caution, as a job ad reflects how an organization wishes to be perceived by these audiences. While job ad analysis alone may not provide a comprehensive description of the discipline in question, we believe that an analysis of the skills and requirements listed in the core “skeleton” of the ads provides a valuable vantage point nonetheless. In this assessment we are not alone, as skill identification from job ads has been gaining popularity in the academic community [4]. A recent survey identified as many as 108 research articles on the subject of skill identification from job ads between 2010 and 2020 [4]. Scientists analyze job ad data to better understand a variety of labour market issues, including how and when companies react to technological change, and hiring discrimination issues [2].

While some studies have sampled job ads across the entire labor market (e.g., [2, 8]), others have focused on jobs in the software engineering and information technology fields (e.g., [1, 3]). Daneva et al. [1] looked at the data from the three most popular Dutch IT job portals over the course of ten months exploring landscape of the requirements engineer specialists. Authors also revealed the most desirable skills and competences for such specialists. Gardiner et al. [3] sampled 1,216 jobs ads which included “Big Data” in their title. The researchers employed computer-aided content analysis and the consensus pile-sort protocol to develop a conceptual model of practitioner knowledge, skills, and abilities expected of Big Data job candidates. The study revealed the prevalence of traditional development skills in such job ads, reflecting that the development of analytics systems is the primary task for many Big Data specialists. The limitations of the study included the single point in time for the data collection, as well as the data being collected from a single job portal skewed towards technical jobs [3].

There are many ways to structure and classify the data collected from job ads. In their working paper on skill requirements across firms and labor markets [2], Deming and Kahn grouped the listed job skills in 10 categories: cognitive, social, character, writing, customer service, project management, people management, financial, computer (general) and software (specific). By identifying the skills expected of Product Ops professionals by the industry, our paper aims

to determine in similar way the profile of Product Ops professionals. The study also assesses how well the above-provided definition encompasses the discipline. To structure and classify the data collected from Product Ops job ads, the 10-category topology of job skills proposed by Deming and Kahn has been adopted [2].

Table 1. Data sources

Job ad platform	Job ads published (“Product Operations”/“Product Ops”)	Job ads selected	Job ads analysed
Startup.jobs	150/15	19	15
Glassdoor.com	1009/149	15	15

2 Methodology

The paper aims to answer the following research question: *What constitutes the profile of Product Ops professionals according to industry expectations?*

To answer this question, we sampled job ads from two job portals: Startup Jobs and Glassdoor. Both portals were searched for job ads containing the keyword “Product Ops” or “Product Operations” in the title. Job ads that included additional term(s) between “Product” and “Ops”/“Operations” were not included, because these titles seem to imply a different focus of the role that might skew the result of the study of “Product Ops” specifically. The study analysed job postings from companies of all sizes and did not consider the geographic location of the companies posting the job openings. The study included Product Ops positions of different levels of seniority. The job ads were selected according to the following inclusion criteria: firstly, the job ads must be written in English, secondly, the job ads must include “Product Ops” or “Product Operations” in the title, and lastly, the job ads must specify working on software-intensive products.

Table 1 describes the number of search results on each job portal, the number of job ads selected, and the number of job ads analyzed during the study. The final sample included 30 job ads. Four job ads initially selected from Startup Jobs were excluded from the analysis because the product described was not software-intensive.

Once collected, the content of the job ads was manually analysed using Nvivo software. Only the “skeleton” of each ad was analysed, including sections such as “About the role”, “What you will do”, “Responsibilities”, “Qualifications”, “The ideal candidate will have”, “What we want to see in you”, and “What your day may look like”. Information about the company such as its history and standing in the market, as well as the benefits it offers to employees, was not included in the analysis. All skills, characteristics, and responsibilities described in the job ad were assigned a code. A degree of subjectivity was exercised in assigning the codes. For example, the following statements were all grouped under a single

“storytelling with data” code: “Experience leveraging qualitative feedback to create product recommendations”, “You will regularly report to the leadership team to review the status of key initiatives, data regarding customer feedback and adoption of products, as well as other key business metrics.”, “Demonstrated experience synthesizing data to craft the narrative”, and “Generating visualizations leveraging different technologies to be included in slide decks and dashboards”.

After the initial coding of the 30 job ads was completed, similar or related codes were grouped according to the 10 categories introduced by Deming and Kahn [2]: cognitive, social, character, writing, customer service, product management, people management, financial, computer (general) and software (specific). While the 10 categories are mutually exclusive, they are not exhaustive [2]. In this study, an additional 11th category “Product Management” was added to address skills and responsibilities that fit within the SPM purview¹. For example, the code “storytelling with data” was grouped under the code “Cognitive”, alongside “problem solver”, “analytical skills”, “excellent judgement”, and others. This grouping resulted in a profile of a Product Ops professional from an industry perspective. The remaining codes were grouped along the four dimensions of the formal definition proposed in [6] and presented in the introduction of this paper. This allowed for an evaluation of the validity of the definition by checking whether there is sufficient evidence in the job ad data to support it. Lastly, the impacts of Product Ops mentioned in the ads were grouped.

3 Results

The 30 job ads reviewed were posted by 29 companies. The most common position in the sample is the “Product Operations Manager” (8 ads), followed by “Director of Product Operations” (4 ads) and “Product Operations Analyst” (3 ads). Two ads look for a “Product Operations Specialist”, and two ads refer to the position simply as “Product Operations”. The remaining job titles were each encountered once: “Tooling Program Manager, User & Product Operations”, “Staff Technical Program Manager – Product Operations”, “Senior Program Manager, Product Ops”, “Senior Data Scientist – Insights (Product Ops)”, “Product Operations Associate”, “Product operations and Applications Manager” (sic), “Product Operations Analyst (Manager)”, “Product Operations Administrator”, “Product Operations (Consultant)”, “Head of New Verticals Product Operations” (sic), and “Head of Product Operations & Delivery Excellence”. Most of the positions are permanent. Only two of the ads specify fixed-term employment.

Almost every ad (98.3%) either requires or prefers candidates with some level of prior experience. Specifically, 63.3% seek experience within the particular software domain the new hire will be involved in, such as fintech, crypto-blockchain,

¹ The ISPMA classifies “product strategy” (including positioning and product definition) and “product planning” (including roadmapping and product lifecycle management) as the core SPM responsibilities [5].

or geospatial products. 36.7% necessitate previous involvement in data-driven decision-making, while 26.7% require former Product Management or Product Ops experience. Preference is given to those with a data-analysis background in 30% of the advertisements. A BSc degree or higher is required by 40%, although some ads accept industry experience as a substitute.

3.1 Product Ops Professional Profile

The recurring codes from the job ad text were classified along the 10 categories by Deming and Kahn [2], plus the added category of “Product Management”. The result is a detailed profile of a Product Ops professional from an industry perspective.

In the “*Cognitive*” category, a Product Ops professional is first and foremost a problem solver (mentioned in 60.0% of the job ads). They possess strong analytical skills (40.0%), and are data-driven (33.3%). They are able to craft stories from data (36.7%), are capable of multitasking (16.7%) and thinking on the spot (16.7%). Product Ops practitioners are comfortable navigating ambiguity (13.3%), are creative (6.7%), demonstrate critical thinking (6.7%) and have excellent judgment (6.7%).

Socially, they are excellent communicators (86.7%) adept at public speaking (23.3%) and possessing interpersonal skills (16.7%). If problems arise, Product Ops professionals know who best to notify, and use appropriate language depending on who they are talking to, for example avoiding technical jargon when dealing with executives (13.3%).

In terms of their *character*, Product Ops practitioners are proactive (46.7%) and detail-oriented (40%). They have a strong ownership mentality of the processes and initiatives they work on (40%), and are curious (30%) and motivated (30%). Product Ops practitioners are able to work independently (26.7%), and are organized (26.7%). They are adaptable (20%) and remain calm under pressure (20%). Ideal candidates are decisive (16.7%) and capable of learning to improve themselves professionally (16.7%). They are also results-oriented (13.3%), accountable (6.7%), and flexible (6.7%). Other traits mentioned include competitiveness, positive attitude, reliability, pragmatism, being an “active listener”, and exuding an “executive presence”.

In the job ads analyzed, 40% explicitly necessitate candidates to possess exceptional *writing* skills, applicable to tasks such as documentation, product training resources, and promotional campaigns. Product Ops professionals are frequently anticipated to engage closely with users and *customers*. 36.7% of the job ads portrays the position as user-centric, while 30% reference working in the interest of the customer. Explicit mentions of direct customer communication are present in 20% of the advertisements, and 10% indicate that the newly hired individual will be accountable for executing a Voice of the Customer (VOC) program.

A significant portion of job ads (33.3%) necessitates that candidates possess *project management* experience, while 30% explicitly require Agile methodology familiarity. Organizational skills are essential for 20% of the roles, with

resource management responsibilities appearing in 6.7% of the ads. Within the “*People Management*” category, a Product Ops expert typically demonstrates cross-functional influence (36.7%), leadership abilities (26.7%), mentoring skills (13.3%), conflict resolution or prevention (10%), and is responsible for staff onboarding (6.7%). Ads call for previous people management experience in 6.7% of cases, with one job ad specifically mentioning involvement in interviewing and hiring. In the “*Product Management*” area, Product Ops professionals engage in product planning (43.3%), which encompasses roadmap development (36.7%) and requirements engineering (10%). They also contribute to product development (20%) and continuous product improvement (26.7%). A smaller proportion of ads (6.7%) highlight the importance of Product Ops experts’ involvement in product strategy development, emphasizing the need for a profound understanding of the products they handle.

A few skills are occasionally mentioned in the “*Financial*” category, such as cost-benefit analysis (6.7%) and capitalization analysis. Some positions mention the candidate would be in charge of ensuring financial compliance (3.3%) and involved in investment planning (3.3%). Product Ops professional must possess a general technical aptitude (“*software (general)*” from [2]) (33.3%), and work regularly with slide decks (16.7%) and spreadsheets (13.3%). Additionally, 33.3% of the ads require the candidate to be familiar with a *specific software* stack, and 30% state a preference for a candidate with sufficient programming skills. In the specific programming languages mentioned, SQL is the most widespread, being mentioned in 30% of the ads. Other languages mentioned are Python, Java, and R. Other programming-adjacent skills listed include familiarity with version control systems, CI/CD pipelines, command line, and low-code tools. Knowledge of software architecture and development life cycles is also mentioned.

3.2 Evaluating the Formal Product Ops Definition

To assess how consistently the representation of the Product Ops role in job ads adheres to the formal definition outlined in [6], the remaining codes identified in the job ad text were grouped along the four dimensions of that definition. Additionally, the impacts of Product Ops were grouped into their own category. The results can be seen in Tables 2 and 3.

In the Data Management dimension, the responsibility of Product Ops experts to gain insights from data is mentioned in 53.3% of the job ads. The involvement of Product Ops professionals in prioritization is mentioned in 46.7% of the vacancies, and their role of decision support is 43.3%. 30% of the ads state that Product Ops professionals strive for simplification within the organization. They are responsible for collecting data (26.7%) and feedback (26.7%), and overall data management (26.7%). Product Ops practitioners are involved in user research (26.7%). They are also responsible for setting up information repositories where everyone in the organization can access the data and resources related to the product (16.7%). Product Ops experts analyze the collected data to improve each iteration of the product, facilitating iterative learning (13.3%).

Table 2. The four dimensions of Product Ops

Dimension	Characteristic	%
Data Management	insights from data	53.3 %
	prioritization	46.7 %
	decision support	43.3 %
	simplification	30.0 %
	collect feedback	26.7 %
	collecting data	26.7 %
	data management	26.7 %
	user research	26.7 %
	setup information repositories	16.7 %
	iterative learning	13.3 %
	experimentation support	10.0 %
	data visualization	6.7 %
	data validation	3.3 %
	Tool & Process Management	process measurement
process measurement \ performance measurement		56.7 %
process measurement \ OKR tracking		43.3 %
process measurement \ roadmap execution tracking		23.3 %
process measurement \ product status tracking		13.3 %
process development		60.0 %
process improvement		60.0 %
drive process adherence		33.3 %
problem identification		33.3 %
tech stack mgmt		30.0 %
internal tool development		23.3 %
internal tool mgmt		23.3 %
automation		16.7 %
obstacle removal		16.7 %
best practice development		10.0 %
Operational Complement		assistant
	provide product support	16.7 %
	troubleshooting support	13.3 %
	let others do meaningful work	10.0 %
Collaboration	communicator	86.7 %
	cross-functional collaboration	83.3 %
	verbal communication skills	43.3 %
	partnership builder	40.0 %
	product status reporting	40.0 %
	documentation	36.7 %
	standardization	23.3 %
	coordinator	20.0 %
	team work	16.7 %
	work with partners	16.7 %
	create cross-functional feedback loops	13.3 %
	keeping documentation up-to-date	13.3 %
	managing cross-functional dependencies	13.3 %
	connective tissue	6.7 %
	product announcements	6.7 %
	team building experience	6.7 %
product demos	3.3 %	

Table 3. Product Ops impact

Characteristic	%
increase efficiency	63.3 %
identify opportunities	56.7 %
help scaling	53.3 %
driving excellence	50.0 %
cross-functional success	46.7 %
drive OKRs	40.0 %
increase clarity	26.7 %
facilitate communication	23.3 %
culture establishment	20.0 %
accelerate feedback loops	16.7 %
feature adoption improvement	16.7 %
increase impact	16.7 %
increase visibility	16.7 %
maximize revenue	16.7 %
empower company	13.3 %
facilitate innovation	13.3 %
inspire	13.3 %
empower SPM	10.0 %
increase transparency	3.3 %
save money	3.3 %

In the Tool and process management dimension, 73% of the job ads state that Product Ops are in charge of measuring company processes to identify opportunities for improvement. Various aspects that are measured include performance measurement (56.7%), OKR tracking (43.3%), roadmap execution tracking (23.3%), and product status tracking (13.3%). 60% of the ads mention process development and process improvement as the core responsibilities, and 33.3% mention that Product Ops drive process adherence. Other responsibilities fitting this dimension include problem identification (33.3%), company and team tech stack management (30%), and internal tool development and management (23%). Automation (16.7%), obstacle removal (16.7%), and best practice development (10%) are also mentioned.

In the Operational complement dimension, the role of Product Ops as assistants to product managers and other company functions is mentioned in 26.7% of the ads. Product Ops help provide product support (16.7%) and troubleshooting support (13.3%). The impact of letting others do meaningful work by reducing the administrative burden and possible obstacles is described in 10% of the ads.

Finally, in the Collaboration dimension, 86.7% of the ads describe the desired candidate as an excellent communicator. A total of 83.3% describe Product Ops experts as the facilitators of cross-functional collaboration. Verbal communication skills are mentioned in 43.3% of the ads. The descriptions of the ideal candidate also depict a Product Ops professional as a partnership builder (40%). They are in charge of product documentation (36.7%) and standardization (23.3%). They act as a coordinator between various company functions, partners, and customers (20%), creating cross-functional feedback loops (13.3%) and managing cross-functional dependencies (13.3%). Teamwork (16.7%) and work with partners (16.7%) are also mentioned.

In terms of their *impact* on the company, 63.3% of the job ads require the Product Ops professional to increase the efficiency of company operations. Many of the ads (56.7%) state the need for Product Ops to identify new opportunities for product and process improvement. Another commonly mentioned impact is to help companies scale and grow (53.3%). Product Ops professionals are supposed to drive excellence and raise the quality bar across the organization (50%), and ensure the cross-functional success of company initiatives (46.7%). They drive OKRs (40%) and increase clarity around all aspects related to the product (26.7%). The desired impact of Product Ops is to facilitate communication (23.3%) and establish a culture of quality and high performance (20%). Other impacts described include the acceleration of various feedback loops, the improvement of feature adoption, an increase of the impact and visibility of company initiatives, and increased revenue. Table 3 contains the full list.

The Product Ops role, as portrayed in job ads, closely conforms to the formal definition proposed in [6]. The statement that “Product Ops [...] makes product companies more efficient and allows them to scale without friction” is certainly reflected in job ads, where “increased efficiency” is mentioned in 63.3% of the postings, and the “help scaling” in 53.3%.

The description of the data management dimension is also supported (“decision support” is mentioned in 43.3% of the ads). However, the “data cleaning” responsibility was rarely mentioned in the job ads – only one ad mentioned “data validation”. It may be implied in the “insights from data” (53.3%), “analytical skills” (40%), and the preferred data-analysis background (30%). The phrase “regularly receives” implies establishing a cadence for communicating with stakeholders, which was alluded to in some of the job ad descriptions and may fall under the “process development” category (60%). The optimization and alignment impact of Product Ops are also largely supported by the quantitative job ad data. “Optimization of time to learn from and react to insights and negative feedback” is reflected in “user-centricity” (36.7%), “iterative learning” (13.3%), and “accelerated feedback loops” (16.7%). The optimization of R&D costs is also alluded to in “prioritization” (46.7%), “identify opportunities” (56.7%), and “maximize revenue” codes (16.7%). The definition of the tool and process management dimension is supported by the “tech stack management” (30%) and “internal tool development and management (23.3%)” codes. The role of Product Ops in process measurement (73%) and improvement (60%) can be further emphasized in the definition, as it is overwhelmingly present in the job ads. The operational complement dimension is present in job ad data (“assistant” – 26.7% and “let others do meaningful work” – 10%), but to a surprisingly small extent. The proactive nature of the job is mentioned in 46.7% of the sources, and the supporting role of Product Ops in relation to other company functions is often alluded to. Finally, the collaboration dimension is largely supported by the job ad data (“communicator” in 86.7%, “cross-functional collaboration” in 83.3%, and “coordinator” in 20%). Product status reporting (40%) and increasing visibility (16.7%) also support the definition.

One aspect that is not explicitly acknowledged in the definition in [6] is the role of Product Ops in ensuring launch readiness of products and features, and management of software releases (described in 30% of the ads). The role of Product Ops in risk analysis on new initiatives is alluded to in 13.3%.

4 Conclusion

In this study, a total of 30 job ads for Product Ops professionals were collected from two job portals. The manual analysis of the job descriptions revealed a professional profile of a Product Ops expert based on the industry expectations for the role. A Product Ops professional is an analytical problem solver who is proactive and user-centric. They excel at communication in all formats and at all levels and can leverage that to exercise a cross-functional influence and build long-lasting partnerships. They can quickly and thoroughly analyze product data and craft a narrative that influences company decisions and moves the needle toward higher efficiency, quality, and revenue. The study also found that the profile of a Product Ops professional that emerged from job ads fits well with the formal definition proposed in [6]. The study recommended expanding the definition with the acknowledgment of the role of Product Ops in process

improvement, ensuring launch readiness, and conducting risk analysis on company initiatives.

The study is subject to limitations. The sample size of the study is small, the data was collected at a single point in time, and the study is only a first step in a more comprehensive analysis of broader swaths of the industry. The numbers included in this paper were provided to clearly describe to the reader the picture that we observed in the analyzed sample. While the exact numbers are likely to change when more job ads are included into the analysis, we expect the list of the skills, qualities, and impacts to remain consistent with the present results.

During the manual content analysis of this study, certain recurring themes were noted but left for further study. One example is the organizational structures of the companies that practice Product Ops. Many of the job titles mention what department the Product Ops hire would be working in, or to whom they would be reporting. There are many ways in which companies structure their Product Ops function [6], and further analysis of this information could illustrate the various company structures.

References

1. Daneva, M., Wang, C., Hoener, P.: What the job market wants from requirements engineers? an empirical analysis of online job ads from the Netherlands. In: 2017 ACM/IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM) (2017)
2. Deming, D., Kahn, L.B.: Skill requirements across firms and labor markets: evidence from job postings for professionals. *J. Labor Econ.* **36**(S1), S337–S369 (2017)
3. Gardiner, A., Aasheim, C., Rutner, P., Williams, S.: Skill requirements in big data: a content analysis of job advertisements. *J. Comput. Inf. Syst.* **58**(4), 374–384 (2017)
4. Khaouja, I., Kassou, I., Ghogho, M.: A survey on skill identification from online job ads. *IEEE Access* **9**, 118134–118153 (2021)
5. Kittlaus, H.B.: *Software Product Management*. Springer Nature, Cham (2022)
6. Moroz, B., Saltan, A., Hyrynsalmi, S.: Product ops: understanding and defining an emerging discipline. In: *GI-Edition: Lecture Notes in Informatics* (2023)
7. Pendo.io. *The Rise of Product Ops*. Technical report, Pendo.io, 2019
8. Rafaeli, A., Oliver, A.L.: Employment ads: a configurational research agenda. *J. Manage. Inquiry* **7**(4), 342–358 (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.



Global and Hybrid Work in Software Engineering



Unveiling the Spectrum of Hybrid Work in Software Engineering: Research Directions

Maria Paasivaara¹ and Xiaofeng Wang²(✉)

¹ LUT University, Lahti, Finland
maria.paasivaara@lut.fi

² Free University of Bozen Bolzano, Bolzano, Italy
Xiaofeng.Wang@unibz.it

Abstract. Despite the heated debate on whether *hybrid work* would be the new normal in the post-pandemic world, it is an exciting, if not new, research phenomenon for Software Engineering (SE) researchers. Hybrid work has a wide range of dimensions and aspects that need exploration and understanding for modern software companies to truly benefit from it. In this paper, we propose a framework that incorporates multiple perspectives on hybrid work in software engineering. We applied the framework to group a set of research topics collected at the GoHyb (Global and Hybrid Work in Software Engineering) workshop collocated with the XP2023 conference, and extrapolated some new topics based on the framework, to demonstrate various research questions that can be asked on hybrid work in software engineering. We conclude the paper with a remark on the need of more contextual and nuanced understanding of hybrid work in software engineering.

Keywords: Research directions · hybrid work · software engineering

1 Introduction

Hybrid work is here to stay, or is it? With hybrid work, people most often mean different combinations of work at the office and elsewhere, e.g., at home. Currently, it seems to be the future trend of all knowledge work and one of the most debatable topics after the pandemic time. One can hear drastically different voices in the media. Some believe that hybrid work represents the best of working from anywhere and working in the office. There are companies, such as LinkedIn, that take hybrid work seriously and re-design their workplaces for hybrid work, optimizing offices for all use cases and accommodating a more diverse workforce [1]. According to a study by the consulting company McKinsey [3], more than two-thirds of the surveyed employees across North America, Europe, and Australia, prefer the hybrid model and claim that they are likely to change their employers if required to return to full-time office work.

On the other side, some claim that hybrid work combines the worst of office and remote work and would call a hybrid work plan “a compromise”. Google went

as far as “crackdown on office attendance” to urge remote workers to adhere to the hybrid work schedule [5]. Recently, it launched “\$99 a night Hotel Mountain View” to attract remote workers to spend time in offices [4]. Meta’s former director of remote work contended that “*hybrid work is not the future... It’s an ‘illusion of choice’*” [11].

The current hybrid work trend and discussion are concerned with all the knowledge work such as software development that does not require a constant physical presence at the office. It is intriguing to see that software companies such as Google, Meta, and Apple are right at the center of this heated debate since hybrid work was arguably more common in software companies than in most other industries before the pandemic time, and one would expect that hybrid work is seen more positively in these companies. In addition, there is already more than decades of experience and a lot of research literature on global software engineering and virtual software teams and there exist plenty of software tools to support remote work. Thus, we believe that software companies and researchers should lead the discussion and research on hybrid work.

The contrasting opinions and arguments from software companies demonstrate that we are yet to reach a good understanding of hybrid work in software engineering which is an exciting research phenomenon and poses distinct challenges compared to what has been already studied in the global software engineering research field. This in turn represents rich opportunities for SE researchers. Research on hybrid work could highly benefit a large number of companies in the field of software engineering that are currently eagerly looking for solutions for their post-pandemic ways of working. Indeed, Conboy et al. [2] urge the need for future research and a research agenda to guide the research community. This paper is our attempt to start drawing such an agenda.

The remainder of the paper is organized as follows. Section 2 proposes a framework for structuring research topics in hybrid work in software engineering. In Sect. 3, we explain how we populated the framework with the input we collected in a research workshop. The results are presented in Sect. 4. We conclude the paper in Sect. 5.

2 A Conceptual Framework to Study Hybrid Work in Software Engineering

A clear definition of *hybrid work in software engineering* seems a prerequisite for creating a research agenda on this topic. Google defines hybrid work as “*a spectrum of flexible work arrangements in which an employee’s work location and/or hours are not strictly standardized*” [8]. According to this definition, besides the location, the working hours of a hybrid worker might be different than his or her co-workers’.

However, as Smite et al. [9] rightly point out, “*the word ‘hybrid’ has become one popular umbrella label attributed to various work-related terms*”, and there is no consensus in SE literature on what exactly *hybrid work* and the related terms mean. This represents one of the central issues to tackle by the researchers. The

desired clarity may be the result of collective research endeavors rather than the starting point. Thus, in this paper, we do not provide a definition but leave that as a future research topic.

It is evident that hybrid work in software engineering is a complex phenomenon, and multiple perspectives are needed in the investigation. We propose a conceptual framework (Fig. 1) with perspectives that can be used to guide the formulation of research questions.

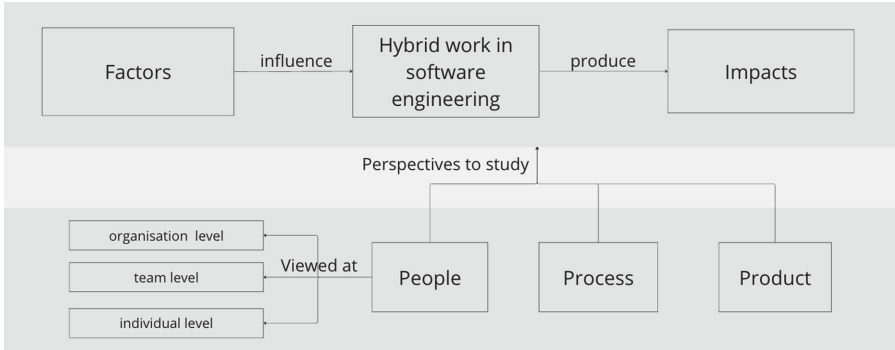


Fig. 1. A conceptual framework for organizing research questions on hybrid work in SE

As shown in Fig. 1, *hybrid work in software engineering* is at the center of the investigation. “Opening this box” means directly researching on how hybrid work is understood and how it is or can be implemented in software companies. Treating it as a “closed-box”, studies can be conducted to either explore what can be the *factors* that influence various policies and implementations of hybrid work or to understand what the *impacts* of hybrid work are in comparison with other formats of work arrangements.

To investigate hybrid work in software engineering, the 3P’s model of software management [6] could be employed. As Reifer [6] argues, for software projects to be successful, *People*, *Process*, and *Product* need to be managed concurrently and conflicts occurring among them be reconciled constantly. Hybrid work, as well as its influencing factors and impacts can be examined along these dimensions. Additionally, the *people* dimension can be further broken down into *individual*, *team*, and *organizational* levels.

3 Research Topics Collection and Organization

3.1 Collecting the Research Topics

We collected research topics on *hybrid work in software engineering* from agile practitioners and researchers through a facilitated workshop session: On June

13th, 2023, we organized a research workshop, *First International Workshop on Global and Hybrid Work in Software Engineering (GoHyb)*¹, collocated with the XP2023 conference² in Amsterdam. This research workshop builds on the Global Software Engineering Conference (ICGSE) and the research and community behind it. The ICGSE conference has been organized yearly since 2000, first as a workshop and from year 2006 onwards as a conference. As hybrid work in software engineering became a new hot topic during the pandemic, we in the global software engineering (GSE) community saw an opportunity to combine the strong research grounds of GSE research and this new industry trend and to organize a workshop on this emerging topic.

In total, 36 persons from around the world gathered in the GoHyb workshop. This half-day face-to-face workshop started with six presentations by practitioners and academics on their practical experiences and research results on hybrid work in software engineering and ended with an interactive session to brainstorm future research topics.

In the interactive session, we used one of the *Liberating Structures* techniques, called *1-2-4-All*³. First, we asked the participants to write individually *future research ideas for hybrid work in software engineering* on sticky notes for a few minutes, then form pairs to discuss, elaborate, and add ideas for five minutes, and finally combine pairs into four-person groups to do the same for ten minutes. In the end, all groups presented their ideas to others: one idea per group, followed by the next group until all new ideas were presented.

3.2 Organizing the Research Topics

After the workshop, we collected the sticky notes and organized them according to the framework we introduced in Fig. 1. Figure 2 presents the result of applying the framework to identify and organize research topics that can be asked on global and hybrid work in software engineering. In Fig. 2, the green sticky notes contain the input we collected from the GoHyb 2023 workshop participants. After organizing the future research ideas from the workshop using the framework, we extrapolated more research topics to complement the input from the workshop especially from the perspective of agile software development. These topics are shown in the blue sticky notes in Fig. 2.

4 Research Topics on Hybrid Work in Software Engineering

In this section, we elaborate on the research topics presented in Fig. 2, starting from the left, *factors that influence hybrid work*, then moving to *hybrid work in software engineering* and how it could be organized, and finally to the *impacts of hybrid work*.

¹ <https://www.agilealliance.org/xp2023/call-for-submissions/global-and-hybrid-work/>.

² <https://www.agilealliance.org/xp2023/>.

³ <https://www.liberatingstructures.com/1-1-2-4-all/>.

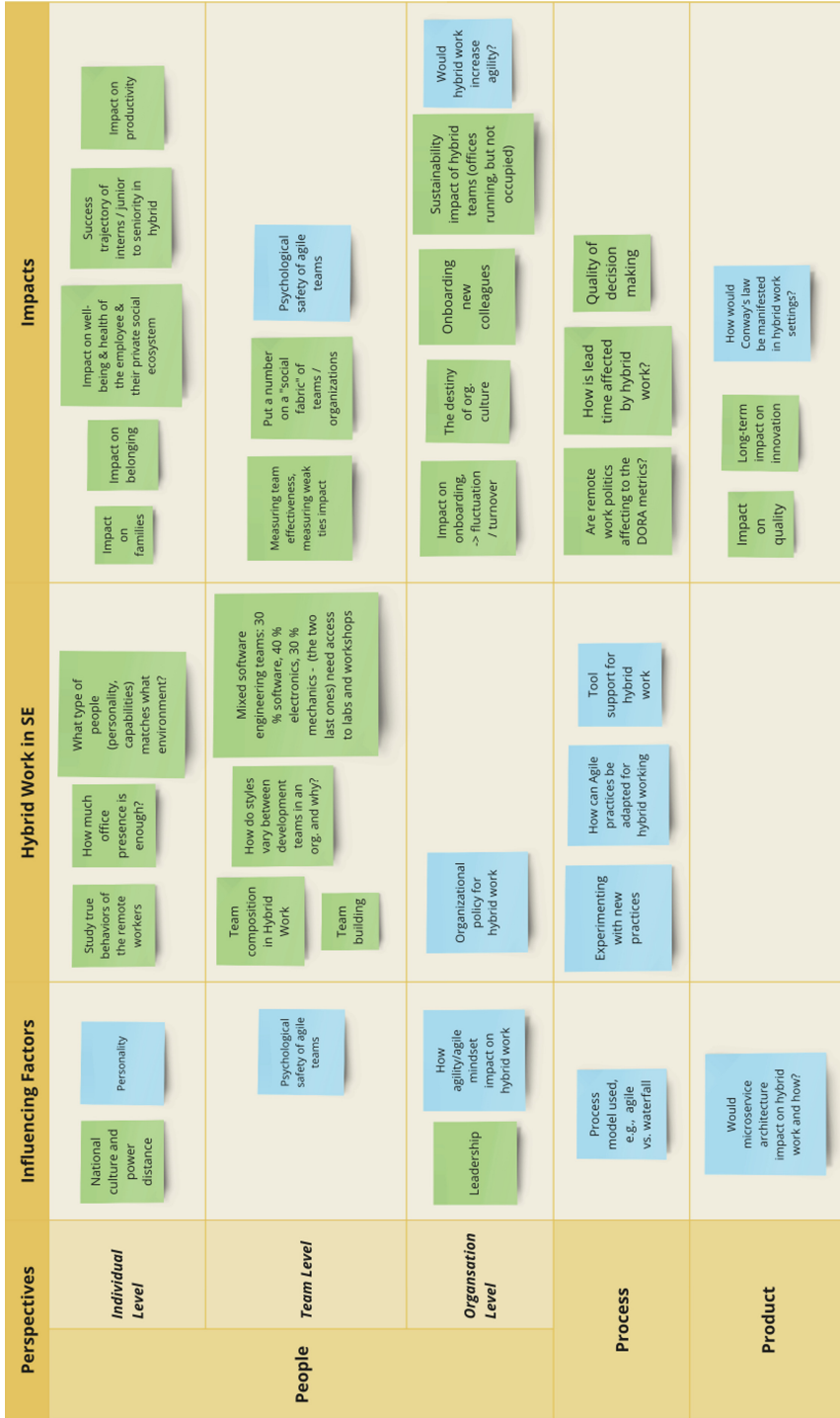


Fig. 2. Illustration of the research topics on hybrid work in software engineering

4.1 Factors that Influence Hybrid Work in Software Engineering

Influencing Factors on hybrid work in software engineering can be examined from people, process, and product perspectives, e.g. whether people or organizations are willing to do hybrid work or whether it fits the product to be developed. As shown by the green post-it notes in the *Influencing Factors* column, we received the least input to this aspect from our workshop participants, however, we added a few topics to give inspiration on what kind of future research ideas this aspect could include. Next, we will elaborate on these.

People Factors. Remote work is not for everyone, and even if it works for a person, it may not be the preferred option all the time. At the *individual level*, understanding what personal factors influence the choice of work mode would be important. *Personality* can be one factor, the *cultural background differentiated by a power distance* could be another one influencing the adoption and implementation of hybrid work.

At the *team level*, how *psychologically safe* the working environment in a team is, may influence how people choose their work mode.

At the *organizational level*, the workshop participants identified *leadership* as an important influencing factor for the effective implementation of hybrid work and raised a question on the *relationship between leadership and hybrid team composition*. When looking at agile software development, an interesting influencing factor might be *how the level of agility or the agile mindset in an organization would influence the hybrid way of working of individual employees and teams*.

Process Factors. Different process models are used in organizations, e.g., agile, waterfall, or a combination of these. An interesting question could be *whether and how the choice of process model would influence how a hybrid work mode is implemented in a company*.

Product Factors. There is a paucity of topics in this category. We encourage SE researchers who have research interests in software projects to consider potential linkages between their research topics and hybrid work in SE. For example: *Would microservice architecture influence hybrid team composition, and if yes, how?*

4.2 Hybrid Work in Software Engineering

People Perspective. At the *individual level*, we could study the *true behaviors of remote workers*, which can be beneficial to dismiss the fear of management losing control when work is not performed at the office, and to enable the management to better support remote and hybrid workers. A relevant question is: *How much office presence would be considered enough?* This links to another interesting question raised by the participants: *What type of people in terms of*

personality and/or capability match different environments in order to be productive?

At the *team level*, the *team composition in hybrid work* is a central topic to investigate: can we create e.g., guidelines to configure hybrid teams? *How do working styles vary between different hybrid development teams in an organization*, and why? *How can team building happen in hybrid mode? How to build high-performing hybrid teams effectively?* Team composition can be an even more complex task to tackle when software development is a part of larger system development. A scenario was described by participants: mixed teams with 30% software, 40% electronics, 30% mechanics people, where the last two groups need access to labs and workshops. How to effectively implement hybrid working with such a team composition?

Even though the workshop participants did not provide any input to the *organizational level*, the other questions they posed point to an important question: *How to define effective organizational policies for hybrid work?* These policies would govern and impact how individuals and teams are allowed and encouraged to work and collaborate, which will be critical when large-scale software development organizations, e.g., several agile teams collaborating on a common product, adopt hybrid work.

Process Perspective. Processes and practices need to be adapted to best support hybrid work. Thus, we could study e.g. *how agile practices can be adapted to hybrid work*. As suggested by Conboy et al. [2], experimentation is important: organizations and teams should *try out what kind of practices work in different hybrid set-ups*, and researchers could collect and report experiences from these experiments. The pandemic already forced us to participate in an experiment: to take into use new tools to support remote work. Many of the tools can support hybrid work as well, and studies on *which tools and how could they best support hybrid work in SE* would be interesting.

Product Perspective. Our workshop participants did not provide any input to this category. Indeed, it is not clear how hybrid work could be studied from a product perspective, or whether this perspective is relevant. It would be interesting to see some future research filling this void.

4.3 Impact of Hybrid Work

Impact on People. At the *individual level*, there are concerns on what would be the *impact of hybrid work on workers' productivity*. SE literature on the productivity of software professionals during the pandemic shows contrasting evidence: some studies show that productivity was not impacted, some show an increase, while others show a decrease in productivity (as reviewed in [10]). However, the long-term tendency and consequences are yet to be seen. The workshop participants raised a concern: *How could young people build their careers and succeed in moving from intern/junior positions to seniority in a hybrid setting?*

Apart from these professional concerns, *physical and psychological health and social well-being of hybrid workers and impact on their families* are prominent concerns that could be investigated further.

The workshop participants posed several questions on the impact of hybrid work at the *team level*. Similar to individual productivity, *team effectiveness* in hybrid settings could be monitored and measured. The participants were concerned about hybrid workers losing connection to others in the organization, especially losing the weak ties, and therefore suggested studying the impact on *weak ties* and on the ‘*social fabric*’ of *teams and organizations*. Additionally, when connecting hybrid work with agile software development, it would be interesting to study how various aspects of agile teams are affected by hybrid settings. Several studies (e.g., [12]) have investigated how the psychological safety of agile teams was affected when they conducted online agile meetings. Similar studies could be conducted in hybrid settings.

At the *organizational level*, one prominent concern is *how a hybrid setting would impact the onboarding of new employees*, which became more challenging during the pandemic time [7]. This is linked to the *turnover of personnel*, as people who are not onboarded properly might leave. Sustainability aspects were brought up as well: *What would happen to offices not occupied? What are the sustainability impacts of these unused resources? How could offices be re-furnished to better fit hybrid work?* The workshop participants also wondered *what would be the destiny of organizational cultures* under the influence of hybrid working. We added a related question: *Would hybrid working have an impact on the agility of an organization?* If yes, how?

Impact on Process. The workshop participants wondered whether remote work politics affect the speed of development (lead time) and the DORA (DevOps Research and Assessment) metrics⁴. A better understanding of hybrid work may also impact on the decision making process, as the input from the participants indicates: *How does hybrid work affect the decision making quality?* Answering such questions can facilitate fact-based decision-making in the context of hybrid work.

Impact on Product. Some workshop participants were concerned with the *quality of software produced by hybrid teams*. Conway’s Law⁵ states that “*any organization that designs a system (defined broadly) will produce a design whose structure is a copy of the organization’s communication structure.*” This could mean that the new communication structures in hybrid work settings would affect the product structure. It would be interesting to study *whether Conway’s law holds also in hybrid work* and we can ask: *How would Conway’s law be manifested in hybrid work settings?* The workshop participants also wondered *what would happen to the software product innovation in the long run*. What would

⁴ <https://dora.dev/>.

⁵ <https://en.wikipedia.org/wiki/Conway%27s.law>.

happen when there are less serendipitous conversations and informal face-to-face communication among co-workers? How should companies support innovation in a hybrid work environment?

5 Final Remarks

The advent of hybrid working at the global scale challenged profoundly our understanding of where, when, and how work can be performed. We did not intend to provide an exhaustive list of research questions on hybrid work in SE. Instead, the collected inputs are exemplar questions serving as an “*enticer*” to tease out more research questions that are worth investigating. The classification of topics/questions under a specific dimension/perspective is not definitive either. Some topics or questions can fit into multiple categories, depending on how they are approached. There are already conducted or ongoing studies covering some of the topics listed in this paper. A systematic analysis of SE literature, or even involving literature on hybrid work in other research fields, could provide a more informed understanding of the landscape of this research area and reveal knowledge gaps to guide future research.

What we hoped to convey with our proposed framework and the illustrative topics is that we do not need sweeping statements with “*a broad brush*” that declare hybrid work as good or bad, as we typically see in the media. What we need are contextual and nuanced understanding of when and how hybrid work could be implemented and could yield benefits. Here is where researchers can play a crucial role and empirical evidence is more valuable than opinions.

It is understandable that the current interest of the industry is to understand the impacts of hybrid work. We expect that, when the impacts are better understood, the interests of both industry and research will shift toward discovering better ways of implementing hybrid work and more proactive actions that will enable hybrid work to produce the desired outcomes.

We wish that more researchers and practitioners would realize that hybrid work is not only about “*where the work is done*” but also about “*how and by whom the work could be done*” in the future. Along this line of thinking, we expect that the meaning of hybrid work in SE and in broader contexts would evolve to embrace richer meanings.

Acknowledgement. We would like to express our sincere gratitude to all the presenters and participants of the GoHyb 2023 workshop for their input, ideas, and discussions. Thank you!

References

1. Britz, L., Norwood, R.: How LinkedIn Redesigned Its HQ for Hybrid Work. Harvard Business Review (2022). <https://hbr.org/2022/10/how-linkedin-redesigned-its-hq-for-hybrid-work>

2. 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>
3. Dowling, B., Goldstein, D., Park, M., Price, H.: Hybrid work: making it fit with your diversity, equity, and inclusion strategy (2022). <https://www.mckinsey.com/capabilities/people-and-organizational-performance/our-insights/hybrid-work-making-it-fit-with-your-diversity-equity-and-inclusion-strategy>
4. Elias, J.: Google is offering an on-campus hotel ‘special’ to help lure workers back to the office (2023). <https://www.cnbc.com/2023/08/04/google-offers-on-campus-hotel-special-to-lure-workers-back-in.html>
5. Elias, J.: Google to crack down on office attendance, asks remote workers to reconsider (2023). <https://www.cnbc.com/2023/06/08/google-to-crack-down-on-hybrid-work-asks-remote-workers-to-reconsider.html>
6. Reifer, D.J.: The “3 P’s” of software management. In: Reifer, D.J. (ed.) *Software Management*, 7th edn (2006)
7. Rodeghero, P., Zimmermann, T., Houck, B., Ford, D.: Please turn your cameras on: remote onboarding of software developers during a pandemic. In: 2021 IEEE/ACM 43rd International Conference on Software Engineering: Software Engineering in Practice (ICSE-SEIP), pp. 41–50 (2021). <https://doi.org/10.1109/ICSE-SEIP52600.2021.00013>
8. Setty, P.: Insights from our global hybrid work survey (2023). <https://workspace.google.com/blog/future-of-work/insights-from-our-global-hybrid-work-survey>
9. Smite, D., Christensen, E.L., Tell, P., Russo, D.: The future workplace: characterizing the spectrum of hybrid work arrangements for software teams. *IEEE Softw.* **40**(2), 34–41 (2023). <https://doi.org/10.1109/MS.2022.3230289>
10. Smite, D., Tkalic, A., Moe, N.B., Papatheocharous, E., Klotins, E., Buvik, M.P.: Changes in perceived productivity of software engineers during COVID-19 pandemic: the voice of evidence. *J. Syst. Softw.* **186**, 111197 (2022)
11. Thier, J.: Hybrid work is not the future, says Meta’s former director of remote work: it’s an ‘illusion of choice’ (2023). <https://fortune.com/2023/07/20/hybrid-work-problems-annie-dean-meta-atlassian/>
12. Tkalic, A., Šmite, D., Andersen, N.H., Moe, N.B.: What happens to psychological safety when going remote? *IEEE Softw.*, 1–9 (2022). <https://doi.org/10.1109/MS.2022.3225579>



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.





Defining a Remote Work Policy: Aligning Actions and Intentions

Darja Smite^{1,2}  and Nils Brede Moe^{2,1} 

¹ Blekinge Institute of Technology, Karlskrona, Sweden

darja.smite@bth.se

² SINTEF Digital, Trondheim, Norway

nils.b.moe@sintef.no

Abstract. After the long period of forced work from home, many knowledge workers have not only developed a strong habit of remote work, but also consider flexibility as their personal right and no longer as a privilege. Existing research suggest that the majority prefers to work two or three days per week from home and are likely to quit or search for a new job if forced to return to full time office work. Given these changes, companies are challenged to alter their work policies and satisfy the employee demands to retain talents. The subsequent decrease in office presence, also calls for transformations in the offices, as the free space opens up opportunities for cutting the rental costs, as well as the other expenses related to office maintenance, amenities, and perks. In this paper, we report our findings from comparing work policies in three Nordic tech and fintech companies and identify the discrepancies in the way the corporate intentions are communicated to the employees. We discuss the need for a more systematic approach to setting the goals behind a revised work policy and aligning the intentions with the company's actions. Further, we discuss the need to resolve the inherent conflicts of interest between the individual employees (flexibility, individual productivity, and well-being) and the companies (profitability, quality of products and services, employee retention, attractiveness in the job market).

Keywords: Flexible work policy · Flexibility · Remote work · Work from home · WFH · Hybrid work · Management · Teams

1 Introduction

In March 2020, in response to the COVID-19 pandemic, most tech companies sent their employees for forced work from home (WFH). Few years later, many knowledge workers still continue voluntary work from home with the preference for at least two or three days per week [1–3]. The reasons for this are manifold, the main being the unwillingness to commute [2], but also simply due to a strong habit of remote work developed during the pandemic [2]. As a result, many consider flexibility a personal right and no longer a privilege [1], and express readiness to quit or search for a new job if forced to return to full time office work [3]. Given these changes, companies are challenged to alter their work policies and satisfy the employee demands for individual

flexibility (often associated with personal well-being and work/life balance [4]) to retain talents [1]. But are individual interests aligned with the team and corporate interests? After all, software engineering is a social activity and focuses on close cooperation and collaboration between all team members [9] and across teams in the organization [10].

The overall goal of our research is to evaluate the state of hybrid working to understand whether employee preferences to work from the office and office presence are changing, whether office presence matters, and if it matters, how can companies encourage employees to return (and how not to discourage the office presence). Despite the rise in research activity on hybrid work, there is still a lot more to be understood about hybrid development to provide rigorous and relevant evidence-based guidance for practice [6]. In this paper, we compare work policies of three Nordic tech and fintech companies that institutionalized flexibility. Our research is driven by the following question:

RQ1: What is the desired office presence in a company?

RQ2: What corporate actions support the desired office presence?

The rest of the paper is organized as follows. Section 2 outlines the key findings from related research. Empirical cases and methodology are presented in Sect. 3. Section 4 is dedicated to the results, and a concluding discussion is found in Sect. 5.

2 Background

Our research of 16 companies and 26 policies in early 2022 demonstrated that WFH policies are divided into three main types: (1) decentralized WFH regulation, (2) centrally regulated onsite/offsite workdays, and (3) centrally regulated proportion of time spent on onsite/offsite, or a combination of these options [1]. Roughly half of 26 studied policies had decentralized WFH regulations with increased levels of flexibility (1), while the other half had centrally regulated onsite workdays, or the proportion of time spent working from home (2 and 3). Few opted for unlimited WFH, similarly to Facebook, Square, Shopify and Slack, who have established policies of long-term and even permanent WFH [6].

Our recent dialogs with the companies suggest that the initial regulations were often related to either the best guesses about the situation or the fears of losing employees or becoming unattractive in the eyes of the new hires, which is also reported in multiple related studies [3]. At the same time, many managers reveal that they prefer employees to return. Such announcements have been received with increasing criticism, as can be illustrated by the exchange of letters between Apple management and employees^{1,2}. Like Apple, many tech companies, even those formally opting for increased flexibility initially, hoped for a gradual increase in office presence. Yet, the actual state of remote work shows that not many people are returning to fulltime (or close to fulltime) work in the office [1–3] and the work has become increasingly hybrid [5]. In contrast to the

¹ <https://www.inc.com/minda-zetlin/apples-remote-work-policy-is-A-complete-failure-of-emotional-intelligence.html>.

² <https://appletogether.org/hotnews/thoughts-on-office-bound-work>.

benefits of being attractive as a workplace and retaining the talents that value flexibility, research started reporting the disadvantages of fully remote and hybrid work [7, 8], returning the considerations about the mandatory office presence. Examples of companies, disillusioned with the actual state of hybrid work, include Apple, Amazon, and Twitter, who pushed for more presence through the formal changes in the work policies. However, research on this topic to date is scarce and largely based on assumptions [6].

3 Methodology

In this paper, we report our findings from studying work policies in three tech and fintech companies, Case A and B (Sweden) and Case C (Sweden and Norway) (see the details of each company in the Findings section). Company names are not disclosed for anonymity. Our research is qualitative in nature and exploratory in purpose. In all three cases, we conducted semi-structured interviews with managers involved in defining the new work policies or responsible for their implementation to discuss how the policies are introduced and supported, and corporate intentions related to office presence (See Table 1). Our findings were discussed with middle managers in each company, and we additionally captured reflections and actions supporting or hindering the corporate strategies. We also visited the premises and captured office observations.

Table 1. Overview of the cases and data collection activities.

Cases	Data sources
A Sweden	<ul style="list-style-type: none"> • Group discussion on remote work policy with five managers (Nov 2022), • Two site visits (one site) (Nov 2022, Feb 2023), • Interview with a manager (Mar 2023), • Office capacity changes
B Sweden	<ul style="list-style-type: none"> • Policy document – “Instruction Remote Work in Sweden” (Dec 2021), • CEO letters to employees (Aug 2021), • Group discussion on remote work policy with the leadership group (Dec 2022), • Group discussion on remote work policies and activities (Apr 2023), • Three sites’ visits (two sites) (Dec 2022, Mar 2023, Apr 2023), • Two interviews with managers (Apr 2023)
C Norway	<ul style="list-style-type: none"> • Policy document – “Future company” (spring 2021), • Two sites visits (Jan and Feb 2022), • Interviews with a manager (Jan 2022, Mar 2023), • Meeting with top management (Dec 2022)

Data analysis was driven by our RQs. We first classified the corporate strategies (Fig. 1) and captured the reasons for the policies (Table 2). Based on the intended office presence, as seen by the management, attitude towards remote work, and necessity for the office space, we identified five corporate strategies, which include: office-based, office-first, hybrid, remote and remote-first (see Fig. 1). We then performed qualitative open coding to identify intentions conveyed in the policy document and the announcements

sent to the employees and corporate actions supporting or hindering these intentions (e.g., in the changes at the workplace). Later, these intentions and actions were mapped to the five corporate strategies and studies for alignment.

4 Findings

4.1 Corporate Policies with Respect to Remote Work

Work policies studied specify the rules that regulate the degree of office/remote work classified into the five possible strategies (see Fig. 1). The five strategies often reflect the intended office presence and the use of office space.

Strategies	Office-based	Office-first	Hybrid	Remote-first	All remote
Intended office presence	Demanded	Encouraged	Fully flexible	Upon request	Not supported
Remote work	Not supported	Allowed	Remote-friendly	Encouraged	Demanded
Necessity for the office space	Available	Sufficient	May be limited	Limited	Limited to none

Fig. 1. Different types of work policies and typical behaviors with respect to intended office presence, attitude towards remote work, and the necessity for the office space.

In the following, we classify the three studied cases according to these five strategies, based on their policies, the announcements made by the management, and by the actions and changes related to the office space, workplace and the other perks offered by the companies. Our findings are summarized in Table 2.

Table 2. Corporate policies, announced and underlying motivations and workplace changes

	Office presence	Remote work	Announced motivation	Underlying motivation(s)	Changes at the workplace
A	Encouraged	Remote-friendly (2–3 days/ week)	Team collaboration	Opportunity to increase cost-efficient use of resources	Downsized office Bookable desks No free parking Better canteen WFH zones at work
B	Encouraged	Remote-friendly (max 50%/ year)	Individual flexibility	Employee retention	No free parking Office-based events Hybrid-friendly meeting rooms

(continued)

Table 2. (continued)

	Office presence	Remote work	Announced motivation	Underlying motivation(s)	Changes at the workplace
C	Encouraged	Flexible, but fully remote is not an option	Team-based flexibility, differences in needs	Employee retention, Engaging and inclusive workplace	Parking is not free anymore, but more places available Office-based events Art classes and sports clubs Better canteen New focused work area, better noise isolation

4.2 Remote work policy in Case A

Case A is a Swedish branch of an international tech company working in the electronics industry. Upon reopening of the offices after the pandemic, the company offices were quite empty with many employees working remotely. The managers explain this with the trend to focus on one's own flexibility first, what they call the "I over We" culture. To change this attitude and to facilitate teamwork, the company decided to implement a hybrid work policy permitting employees to work from home 2–3 days per week. The motivation behind the mandatory office days, as the manager explained, is related to the needs of the teams:

"Team days are important. [...] Some tasks take a bit longer time if you do them digitally, it is preferred to meet when you have problem solving and creative tasks, since doing them on a distance requires other skills from a leader to manage. Finally, decision-making requires presence".

To further support the return of the employees to the office, it was decided to negotiate the change of the canteen with the landlord, and even influence the choice of the menu on certain days. However, the policy in Case A is not strictly monitored, and the offices still have not been filled. In fact, many employees were reported to have a feeling of sitting in an empty office. Due to this reason and because of the relocation of some units to a new office and the end of a rental contract, the company decided to downsize the office by reducing the space from four to two floors. This has increased the density of the employees and made the environment seem more social. Interestingly, unlike many other companies, this has not resulted in the limited space. As a manager explains:

"We have downsized, made the office smaller, and we have also renovated and changed some areas. But we have seats for all, it is still possible to fit everyone. The downsizing primarily affected the unused space".

One potential factor that perhaps did not support the return of the employees to the offices is the renegotiated conditions for the parking lot, which cancelled the free parking and free charging stations for electrical vehicles. However, the employees were said to not complain and to understand the reason for the cost.

4.3 Remote Work Policy in Case B

Case B Sweden is one of the sites of a large international company delivering software-intensive systems to the telecommunication market that employs around 500 people in the studied corporate site. After the pandemic, the company decided to focus on increasing the office presence. For long, there have been no renovations or downsizing in the office, and there is still space for everyone. Corporate and site management repeatedly emphasized the importance of helping and supporting colleagues over focusing entirely on one's own tasks, as well as innovating and driving the corporate culture. These intentions are largely rooted in a belief that some tasks cannot be done effectively when everyone works remotely. However, office presence after the pandemic is not high, especially in larger cities.

When it comes to the policy, the introduced remote work rules are very broad and demand employees to work from the office *“at least 50% of the time during a calendar year”*. Many agree that the policy is very vague and hard to follow, as a manager explains – *“It is not controlled, nobody is measuring one's office presence”*. The underlying motivation for the policy can be found in the Swedish legislation – if an employee spends 50% of time or more working from home, the employer is responsible for the equipment and ergonomics of the home office. Therefore, the policies like in Case B may be falsely understood as unwillingness to take the responsibility for the employees' well-being while they are working from home.

Due to the half-empty offices, the free parking deal was cancelled, and the office space will soon be reduced. More costly commute to work in Case B is a larger problem than in Case A, since most workers commute to work by car. Further, in another company location in Sweden the downsizing will result in closing down an office building, relocating the units into other buildings and having limited seating for the employees. This might be problematic, since the office presence seem to be increasing. As a manager explains:

“We have seen an increase in office presence in the last months. [In four weeks] it went from around 34% to about 47% [...] We can feel this in our parking area”.

The increase in the office presence is probably the result of numerous activities taken by the management and the employees themselves. Teams were said to organize team breakfasts, while the management initiatives include few weeks long *“return to the office”* events, onsite seminars, gatherings and afterworks.

4.4 Remote Work Policy in Case C

Case C is a financial services company headquartered in Norway, which operates in the Nordic markets. The company employs more than 2,000 people in total. During

the pandemic employees in the bank reported many benefits of working from home [4], including an increased ability to focus, fewer distractions, increased flexibility to organize one's work hours, less time spent on commuting, as well as more efficient and shorter meetings. Subsequently, many wanted to continue working from home after the pandemic. To satisfy the individual needs and maintain low retention the company introduced a very flexible work policy with only one exception – fully remote is not an option. Further, the policy minimized business travel (to meet the company sustainability goals), which was also used as one motivation for why not to allow employees to live on a far distance.

After reopening of the offices, employees started returning. The management discussed whether to introduce rules for mandatory office days. However, it was decided against the centralized regulation because different tasks have different requirements and a work unit (team, group, department) has much more insights for such decisions. As an HR manager explains:

“We need to understand that different employees and functions have different needs! [...] One must also not misunderstand the approach as an anarchy where everyone has to decide for themselves. [...] Our strategy as such we call Team-based flexibility with office as the core.”

The company management does not believe in a one-size-fits-all or that there is one, lasting solution that a team can have. The need for regular discussions is emphasized similarly to the way the goals, processes and tools are discussed on a regular basis. Albeit, achieving an agreement in teams with diverse preferences for onsite/remote work is not an easy task. The company is currently working towards understanding how to balance the needs of the individual vs the team vs the company, and how to account for the changes in the context, for example, when a team onboards a new member. The HR manager continued:

“We know that we have highly educated employees who are responsible for complex processes and solutions. Our social mission is, among other things, to ensure that our customers have financial security and freedom. With this as a backdrop, we believe it is unwise to use too many rules and policies to try to manage the organization.”

Case C strategy is not to force employees back but to offer them attractive conditions to return, including better lunch, office-based events on Tuesdays, afterwork events, art classes and sport activities. Further, as the company became aware of the importance of uninterrupted work, they rebuild the offices introducing noise cancelling textures, furniture and walls, and created several quiet zones.

5 Concluding Discussion

In this paper, we studied the institutionalized degree of office/remote work in three Nordic companies, and the actions and changes in the office implemented after the pandemic and how these support the intentions. See the summary of our results Table 3.

Our findings suggest that the studied companies have similar intentions, but different approaches to regulating office presence. All three companies believe in the importance of office collaboration and express the emphasis on the office similarly (Office as “The main place of work” in Case A, “The center of innovation, learning and driving the culture” in Case B, and “The core” and “The base” in Case C). Yet, while Case A and B have introduced minimum demands on the office presence similarly to such tech giants as Apple, Amazon, and Twitter, Case C only states that working fully remote is not an option. Ironically, the company with the most flexible policy seem to have succeeded to attract more people back. Our findings suggest that one reason for this is related to the corporate actions that “lure” the workers back, and the way of doing hybrid must be adjusted to the people and tasks as also suggested in earlier research [6, 9]. The actions in all three companies largely overlap, with some differences in the impact on the workers (for example, cancelled free parking in Case B had a larger impact than in Case A), and some additional amenities in Case C.

In Table 3, we visualize the “messages” that employees in all three companies “receive” through the corporate policies, management announcements and the actions introduced by the companies, including workplace transformations and renovations, changes in the amenities and onsite events. It is evident, that no one company is fully consistent in their intentions. In the following, we list the discrepancies and competing interest that companies shall take into consideration to set informed priorities:

- **Discrepancy: Intention to increase office presence while cutting the workspace.** Maintaining half-empty offices is not a cost-efficient strategy and many companies downsize their offices or onboard more people without increasing office capacity [11]. In doing so, some employees are destined to less convenient working (no personalized space, unpredictable seating) or shortage of work desks, which affects the office presence negatively. Important software development practices like pair programming have been found to suffer in such conditions [12] because developers are disturbing each other.
- **Discrepancy: Intention to increase office presence while removing free parking.** Along with downsizing, many companies cancel free parking deals, resulting in the less convenient and more costly commute, which again may negatively affect office presence or employee satisfaction.
- **Competing interests: Intention to increase office presence while keeping the individual workers satisfied.** These competing corporate and individual interests are hard to satisfy since many workers prefer to work increasingly remotely [1, 3, 6]. In our study, these competing interests often manifest in discrepancies in how management communicates the desired office presence (See “Announcements” in Table 1) and how it is formally regulated (See “Policies” in Table 1).
- **Competing interests: Intention to satisfy individual and team needs.** Finally, the interests of the teams and individuals might not be aligned, especially when the team is composed of workers with different preferences for onsite/remote work. Our study shows that coming to an agreement about suitable flexible working rhythm in a team is not always an easy task, but promotion of team values and team-based decisions sets clear priorities.

Table 3. Discrepancies in intended office presence in policies, announcements and actions.

	Presence	Policies	Announcements	Actions
A	Quite stable, 46% (average)	[Hybrid]: New work policy is remote-friendly, teams (not individuals) decide.	[Office-first]: Office presence is encouraged, Office is the main place of work, some tasks require presence	[Support for office work]: Cozy WFH zones, better canteen. [Support for hybrid work]: Free seating, no personalized desks at the office. [Support for remote work]: Free parking and electrical charging stations were cancelled.
B	Recently increased from 34% to 47% (average)	[Hybrid]: 50% of the work time during a calendar year shall be spent onsite.	[Office-first]: Office presence is encouraged; Office is the center of innovation, learning and driving the culture [Hybrid]: Flexibility is here to stay; we will offer that.	[Support for office work]: Onsite events. [Support for hybrid work]: Hybrid-friendly meeting rooms. [Support for remote work]: Free parking deal and electrical charging stations were cancelled. Ongoing and planned downsizing projects.
C	Quite stable, 55% (average)	[Hybrid]: Fully remote is not an option, teams decide on the rhythm.	[Office-first]: Office is encouraged; Office is our base; Presence is needed for coincidental encounters. Our strategy is team-based flexibility with office as the core. [Hybrid]: Getting people back is not the goal. We aim to facilitate inclusive hybrid working.	[Support for office work]: Onsite events, courses, sports activities. [Support for hybrid work]: Hybrid-friendly meeting rooms. Financial support for the home office

We conclude that hybrid work is prone to inherent conflicts of interest between the individual employees (flexibility, individual productivity, and well-being), the teams (effective collaboration, spontaneous interaction) and the companies (profitability, quality of products and services, employee retention, attractiveness in the job market) that require attention when formulating the new work policies. One step towards understanding whether the intentions and actions are aligned is to perform a mapping based on the corporate strategy documents, announcements and revising the changes at the workplace, similar to the one offered in this study. Resolving the potential conflicts is not an easy task and requires a clear motivation behind the chosen work policy.

Acknowledgements. This research is funded by the Research Council of Norway through the 10xTeams project (grant 309344) and the Swedish Knowledge Foundation through the KK-Hög project WorkFlex (grant 2022/0047).

References

1. 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)
2. Smite, D., et al.: Jyväskylä. Finland **2022**, 252–261 (2022)
3. Barrero, J.M., Bloom, N., Davis, S.J.: Let me work from home, or I will find another job. In: Working Paper, University of Chicago, pp. 2021–2087 (2021)
4. Smite, D., Tkalic, A., Moe, N.B., Papatheocharous, E., Klotins, E., Buvik, M.P.: Changes in perceived productivity of software engineers during COVID-19 pandemic: the voice of evidence. *J. Syst. Softw.* **186**, 111197 (2022)
5. Smite, D., Christensen, E.L., Tell, P., Russo, D.: The future workplace: characterizing the spectrum of hybrid work arrangements for software teams. *IEEE Softw.* **40**, 34–41 (2022)
6. 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)
7. Yang, L., et al.: The effects of remote work on collaboration among information workers. *Nat. Human Behav.* **6**(1), 43–54 (2021). <https://doi.org/10.1038/s41562-021-01196-4>
8. de Souza Santos, R.E., Ralph, P.: Practices to Improve Teamwork in Software Development During the COVID-19 Pandemic: An Ethnographic Study. In Proceedings of the 15th International Conference on Cooperative and Human Aspects of Software Engineering, 2022, pp. 81–85
9. Mens, T., Cataldo, M., Damian, D.: The social developer: the future of software development [guest editors' introduction]. *IEEE Softw.* **36**(1), 11–14 (2019)
10. Berntzen, M., Hoda, R., Moe, N.B., Stray, V.: A taxonomy of inter-team coordination mechanisms in large-scale agile. *IEEE Trans. Softw. Eng.* **49**(2), 699–718 (2022)
11. 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>
12. Tkalic, A., Moe, N.B., Andersen, N.H., Stray, V., Barbala, A.M.: Pair programming practiced in hybrid work. In: A Short Paper in the International Symposium on Empirical Software Engineering and Measurement (ESEM) (2023)

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.



Fear-Based Agile Transformations



Business Development in Large-Scale Agile Software Development: Barriers and Enablers

John Olav Olsen¹ , Viktoria Stray^{1,2}  , and Nils Brede Moe² 

¹ University of Oslo, 0373 Oslo, Norway
stray@ifi.uio.no

² SINTEF, Trondheim, Norway

Abstract. Currently, many financial organizations must undergo a digital transformation. In this study, we investigated a transformation in a Norwegian fintech company with the aim of understanding how the tasks performed by business development can be better aligned with the work of cross-functional development teams. Specifically, we examined the enablers and barriers to coordination between business development and software product development in large-scale agile software development. The organization under study had 25 software product development teams that followed an in-house agile model. We collected data by conducting 13 interviews and collecting various documents. Our findings suggest that having cross-functional fora, having a common understanding of what business development is, and coaching the whole organization to be more agile can improve coordination between business and software development.

Keywords: Collaboration · Alignment · Culture · Strategy · Coordination · Continuous software development · Self-managing teams · Autonomous teams · Digitalization

1 Introduction

Software product development and business development (BD) in large-scale agile requires closer collaboration between actors such as legal representatives, customer service agents, market and business representatives, designers, developers, testers, and maintainers. However, contemporary research shows that software development has been characterized by harmful disconnects between important activities such as planning, development, and implementation. Therefore, the link between business strategy and software development needs to be addressed and improved [10, 19].

Even if there seems to be a broad consensus in the literature that a holistic approach to software development is needed [8, 10, 11], at the same time, topics related to BD are often found outside the scope of software development as well as outside the cross-functional product development teams.

In large-scale agile, it is crucial for business and software product development teams to coordinate well, which can be challenging due to the size

[4, 6, 9, 14]. Effective coordination of work within large-scale agile software engineering is key to project success, and researchers have addressed topics related to leadership, organizational context, design of teams, autonomy, and team processes [7, 13]. Further, team autonomy must be balanced with the larger organizational structures because of a need for alignment between the system, the organization, and the product [5, 6, 15]. Challenges in large-scale agile include integrating non-development functions, change resistance, stakeholder management and keeping to the agile principles [6, 8, 9].

Berntzen et al. [3] found 27 coordination mechanisms across three categories in their study of 24 teams: Meetings, roles, tools, and artifacts. They found that the product managers, development managers, and customer managers were important for managing business process dependencies. Further, Bass emphasized the functions [1] and activities [2] performed by product owners to demonstrate the significance of the role for inter-team coordination.

We aimed to explore the overall research problem of balancing and aligning BD and product development in large-scale software development by investigating the following research question:

RQ: *What are the barriers and enablers for coordinating business development (BD) and software product development in large-scale agile?*

To answer our research question, we conducted a case study of a Nordic fintech company, hereafter called SoftCo.

2 Context and Methodology

Table 1. Overview of the interviews

No.	Role	Comp. Exp.	Years Exp.	Duration
1	Agile Coach	3	15–20	62 min
2	Product Manager	6	10–15	53 min
3	IT Manager	3	20–25	61 min
4	Business Developer	4	30+	57 min
5	Sales Manager	4	10–15	66 min
6	Business Developer	3	20–20	57 min
7	Business Developer	5	30+	54 min
8	IT/Tech Manager	4	10–15	55 min
9	Sales Manager	5	10–15	55 min
10	Sales Manager	5	10–15	63 min
11	Product Manager	5	10–15	59 min
12	Product Manager	5	10–15	53 min
13	Product Manager	4	15–20	70 min

SoftCo started as a service from a large Nordic enterprise and was later spun off as a separate company. SoftCo offers a payment infrastructure for the Nordic market, and operates in both the B2B (business-to-business) segment and the B2C (business-to-consumer) segment. The software development is done almost completely in-house. During the company’s lifetime of approximately five years, there have also been mergers and acquisitions, with the consequence that the existing code-base of SoftCo’s products may have different origins. The company culture is based on agile values, with a focus on flexibility and autonomy.

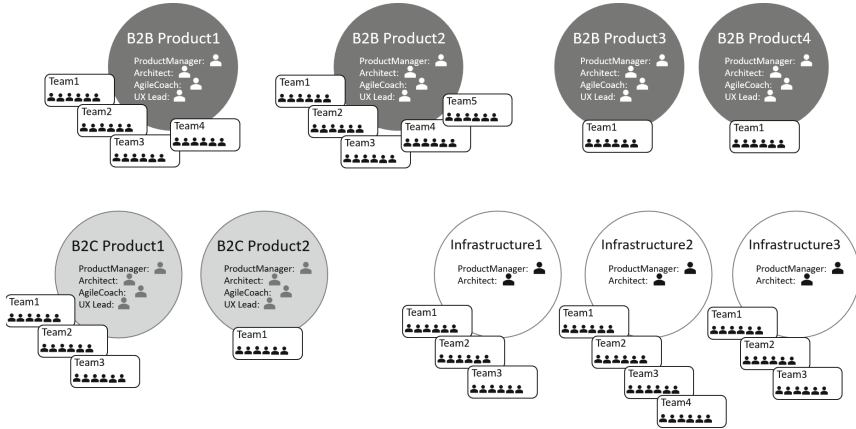


Fig. 1. Product areas and their associated teams

The products are separated into three product areas; *B2B* - serving business customers, *B2C* serving end-users as consumers, and *infrastructure products* covering products related to payment infrastructure (see Fig. 1). Each product area is managed by an area product manager, with several underlying product managers and product teams. These cross-functional teams are often referred to as *product- and tech teams*, with team members such as an Engineering Manager (tech and personnel responsibility), a Product Manager (OKRs and P&L responsibility), UX and Interaction Designers (usability risk responsibility), back-end developers, and front-end developers. The product areas, the products, and the 25 cross-functional development teams are shown in Fig. 1. In addition, SoftCo has several other teams with more commercial-oriented focus, such as Marketing, Sales, Business Development, Strategy and Finance, and finally, Legal and Compliance. The employees within Business Development, Strategy, and also, to a certain degree, within the Sales unit label themselves as business developers. Much of their work is related to gaining new income by increasing market positions and developing partnership models. These teams are hereafter called *commercial teams*.

Slack was the main communication platform for both direct communication, daily group communications, and sharing documents. There are currently more

than 1600 channels in use, and all of the company’s approximately 300 employees and consultants are able to create a channel. Open channels are highly recommended. A statistical report for a 30-day period shows that more than 243.000 messages were written in Slack, where 40% in public channels, 40% in direct messages, and 20 % in private channels.

Based on the research questions of this study, a qualitative approach with a descriptive and interpretive design has been chosen. The data collection has been done through in-depth interviews and document analysis. Such kind of a case study is explained by Yin (p. 5) [20] as an in-depth investigation of a real and contemporary phenomenon in a real-life context. The interviews were conducted in December 2021–March 2022 and lasted, on average, one hour; see Table 1. All recordings were transcribed, and we used NVivo to analyze the interviews and the Ladder of Analytical Abstraction [12] to structure the data analysis. This method helped us categorize, sort, and find patterns to reduce raw data. A predefined coding scheme was a starting point, with room for data-driven coding and grouping into new categories and themes.

3 Results

It is one thing to focus on continuous processes and trying to bridge the gap between commercial activities and product development; it is quite another to enable such collaboration and coordination to work in practice. Here, we focus on the main barriers and enablers in our empirical study on the team and organizational level when engaging in large-scale agile product development.

3.1 Barriers

Unclear and Ambiguous Understanding of BD. Our analysis showed that people in the commercial and product teams had different understandings of the role of business developers and the tasks performed by a business developer. Because of missing role clarity (clarity employees have about the requirements and tasks for their work and others’ work), alignment between business and product teams became problematic.

When analyzing what the interviewees understood as business development, we found six different perspectives; see Table 2 for a summary. For example, one of the interviewees stated that business development and software product development *“are the same thing, and there should not be anyone else than the product department that should work with such topics”*. The interviewee meant that many resources may give their input, but it is finally the product managers who decide what to do. Other interviewees said there are many similarities between product and business development, and the line between the two is often unclear. Another perspective was that business development includes developing products, but goes wider and broader. A product manager explained that *“Business development is the level above product development, because it also includes*

Table 2. Different Perspectives on Business Development

Perspective	Description
1. Developing new products	BD and product development are the same thing, and only product managers should work on such topics. Other units may contribute, but they have no execution power
2. Supporting sales and management	Perception of BD as similar to project management
3. Creating new business on existing products	BD is more about creating new business on existing products, identifying market needs, and using sales and marketing resources to serve the market with existing products and services
4. Creating value through partnerships	BD is about creating value through partnerships and conquering new market positions by selling existing products to new markets
5. Finding customers, markets, and distribution channels	BD includes developing products, but it goes wider and broader, such as finding customers, markets, and distribution channels for the product being developed. It also relates to future possibilities. Balancing short-term and long-term goals and anticipating future market needs
6. Creating value for the owners	BD is about creating value to the owners of the company through creating financial income, creating products and services, creating a market, and creating channels to sell those products and services. Pricing structures and business models are vital parts of BD, and it needs to be included in several business parts of the company

a holistic approach, such as finding customers, markets and distribution channels for the product being developed”.

These different perspectives indicate that having a unified definition of BD could reduce barriers. Based on our findings, we suggest that a definition of BD should include activities for creating new value, by creating new market positions, establishing new distribution lines, creating new products or new product features, or winding down the product portfolio to focus on other market positions.

Us Versus Them Culture. The commercial teams experienced that it was difficult to provide input on business development aspects (such as how to strengthen the market with new products and features) to the product- and tech teams. Some interviewees described that one reason was that the product teams wanted to have a bottom-up culture, be autonomous, and decide tasks themselves. For many, it felt like an *us versus them* mentality between the commercial teams and the product teams. The input from the commercial side was

treated on many occasions with the same resistance that a top-down management approach would have received. One of the sales managers said:

“The development teams have no deadlines. They can deliver whatever they want at any time, and we struggle to give our input to this process.”

The mentality was a barrier that reduced collaboration between business developers and product teams. A business developer stated, *“At worst, those teams think they are autonomous, so they will deliver something they believe is important, but in reality, there are other understandings from other parts of the organizations of what should be the most important thing to deliver.”*

Agile only Embraced by Parts of the Organization. The product- and tech teams had used agile methods and techniques from the beginning. Trying to make a distance to the origin companies that SoftCo had spun out from, they had chosen not to entirely go for *a well-established large-scale agile framework*, but rather develop an in-house model, based on agile principles with a high degree for flexibility and autonomy for the product- and tech teams. This in-house model was well documented, and all interviewees explained that it was known to the whole company that “this is how the company executes their product development”. However, the commercial teams, including business developers and strategy resources, had their own processes. Those methods were not documented in the same extensive way as the in-house agile development model of the product- and tech teams, and some interviewees explained that they did not even know what the *“so-called business developers”* were doing. One product manager proclaimed:

We [the product-and tech teams] follow the company development model for product development, while “they” [BD] just follow their gut feeling

The lack of discussion of how to work together and what common processes to follow divided the company into two, which then strengthened the us versus them mentality. Since common processes, principles, and tools were missing, synchronization, alignment, and coordination of work became problematic. When interviewing people, many argued that it would be difficult to have a one-size-fits-all methodology, mainly because BD is both a *broader* and perhaps more *future oriented* area compared to product development focusing more on short term perspectives.

3.2 Enablers

Cross-Functional Business Fora. The cross-functional business fora, with bi-weekly meetings, helped bridge the gap between the commercial and product teams. Those fora were open to more people across the units, so they became an important arena for cross-functional discussions, with a wider professional coverage compared to what was found in the product- and tech teams. The foras

was seen as a mean to prevent silo thinking, and the members of the commercial teams felt included. Further, the dialogue and dynamics that evolved in those fora reduced the *us versus them* mentality.

Having Agile Coaches. The in-house agile development model was developed and managed primarily from the product development side of the organization. The agile coach interviewed said they would like to spend more time educating and informing the whole organization about how to use agile methods. Several interviewees indicated that having agile coaches helped reduce the “*us versus them*” mentality and the use of statements such as “*our model*” versus “*their model*”. The commercial and product teams gave characteristic and polarized descriptions of each other and each other’s working practices. The agile coach understood both the business and software development perspectives and functioned as a valuable intermediary between the two groups.

A Unified Strategy and Collaborating on Goal-Setting. Our findings indicate that having a strong and clear strategy helped align the autonomous teams and prevented them from running in different directions. Also, in SoftCo, some used Objectives and Key Results (OKRs), which helped increase harmony between the commercial teams that were working with long-term strategic BD, and the product teams working on concrete tasks for the sprint. It also made it easier to collaborate on BD activities and helped the commercial and technical sides work together towards a common goal.

4 Discussion and Conclusion

In this study, we explored the challenges and enablers of aligning business development and product development in large-scale software development by asking, “*What are the barriers and enablers for coordinating business development and software product development in large-scale agile?*”

Our findings confirm previous research that coordinating business and software product development teams is challenging in large-scale agile [4]. Our case study revealed that an unclear understanding of BD and business developers’ roles and responsibilities hindered alignment between commercial teams on one side and the product teams on the other. An unclear terminology coupled with an “*us versus them*” mentality, created barriers for collaboration and reduced the effectiveness of business development input. As a common understanding of terms is important, based on our case study, we suggest the definition of business development in the context of large-scale agile software development as shown in Fig. 2.

Furthermore, the absence of common processes and principles between agile product teams and more commercial-oriented teams perpetuated this divide, making synchronization and coordination of work challenging. Addressing these issues may require tailored agile methodologies that promote shared understanding, collaboration, and integration of processes throughout the organization. Our

Business development definition

Business development refers to the processes and activities that aim to grow and expand a business. Such activities include: creating new market positions, establishing new distribution lines, creating new products or features, and winding down the product portfolio to focus on other market positions.

Business development can be performed by several parts of an organization, both inside the agile product development teams and in other units such as sales-, market- and strategy teams. Who is involved depends on the product's life cycle stage.

Fig. 2. Suggested definition of BD

findings suggest that agile coaches can help reduce the “us versus them” mentality by educating the entire organization. This finding supports that agile coaching should not be limited to the team level and that an important part of the work of a coach is to facilitate overcoming human-related obstacles and to guide both stakeholders and managers in the implementation of agile methods [17].

We found that the use of OKRs also helped improve the coordination between commercial teams and product- and tech teams, where people from business development and software product development worked together to agree on common future objectives. These fora can be understood as communities of practice or guilds [16]. For example, commercial teams worked together with the product teams when setting OKRs for the respective product area, and this seemed to have a positive effect in building a clear strategy. The use of collaboration tools like Slack and goal-setting frameworks such as OKRs has earlier been shown to play a vital role in coordination in large-scale agile where Slack enabled frequent, timely, and problem-solving communication, and OKRs facilitated knowledge sharing, goal alignment, and inter-team coordination [18].

Bridging the gap between commercial teams and agile product teams requires a reorientation not only by developers and business developers but also by management. Making such changes takes time and resources, but it is a prerequisite for the success of any kind of large-scale agile product development.

Acknowledgments. We would like to thank the studied company for their engagement in our research. The work was partially supported by the Research Council of Norway through the projects 10xTeams (grant 309344) and Transformit (grant 321477).

References

1. Bass, J.M.: How product owner teams scale agile methods to large distributed enterprises. *Empir. Softw. Eng.* **20**, 1525–1557 (2015)
2. Bass, J.M., Haxby, A.: Tailoring product ownership in large-scale agile projects: managing scale, distance, and governance. *IEEE Softw.* **36**(2), 58–63 (2019)

3. Berntzen, M., Hoda, R., Moe, N.B., Stray, V.: A taxonomy of inter-team coordination mechanisms in large-scale agile. *IEEE Trans. Software Eng.* **49**, 699–718 (2022)
4. Berntzen, M., Stray, V., Moe, N.B., Hoda, R.: Responding to change over time: a longitudinal case study on changes in coordination mechanisms in large-scale agile. *Empir. Softw. Eng.* **28**(5), 114 (2023)
5. Bick, S., Spohrer, K., Hoda, R., Scheerer, A., Heinzl, A.: Coordination challenges in large-scale software development: a case study of planning misalignment in hybrid settings. *IEEE Trans. Software Eng.* **44**(10), 932–950 (2017)
6. Dikert, K., Paasivaara, M., Lassenius, C.: Challenges and success factors for large-scale agile transformations: a systematic literature review. *J. Syst. Softw.* **119**, 87–108 (2016)
7. Dingsøy, T., Falessi, D., Power, K.: Agile development at scale: the next frontier. *IEEE Softw.* **36**(2), 30–38 (2019)
8. Dingsøy, T., Moe, N.B., Fægri, T.E., Seim, E.A.: Exploring software development at the very large-scale: a revelatory case study and research agenda for agile method adaptation. *Empir. Softw. Eng.* **23**(1), 490–520 (2018)
9. Edison, H., Wang, X., Conboy, K.: Comparing methods for large-scale agile software development: a systematic literature review. *IEEE Trans. Software Eng.* **48**(8), 2709–2731 (2021)
10. Fitzgerald, B., Stol, K.J.: Continuous software engineering: a roadmap and agenda. *J. Syst. Softw.* **123**, 176–189 (2017)
11. Leffingwell, D.: *Scaling Software Agility: Best Practices for Large Enterprises*. Pearson Education, New York (2007)
12. Miles, M.B., Huberman, A.M.: *Qualitative Data Analysis: An Expanded Sourcebook*. Sage, Thousand Oaks (1994)
13. Moe, N.B., Stray, V., Hoda, R.: Trends and updated research agenda for autonomous agile teams: a summary of the second international workshop at XP2019. In: Hoda, R. (ed.) *XP 2019*. LNBP, vol. 364, pp. 13–19. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-30126-2_2
14. Paasivaara, M., Behm, B., Lassenius, C., Hallikainen, M.: Large-scale agile transformation at Ericsson: a case study. *Empir. Softw. Eng.* **23**, 2550–2596 (2018)
15. Rigby, D.K., Sutherland, J., Noble, A.: Agile at scale. *Harv. Bus. Rev.* **96**(3), 88–96 (2018)
16. Smite, D., Moe, N.B., Floryan, M., Levinta, G., Chatzipetrou, P.: Spotify guilds. *Commun. ACM* **63**(3), 56–61 (2020)
17. Stray, V., Memon, B., Paruch, L.: A systematic literature review on agile coaching and the role of the agile coach. In: Morisio, M., Torchiano, M., Jedlitschka, A. (eds.) *PROFES 2020*. LNCS, vol. 12562, pp. 3–19. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-64148-1_1
18. Stray, V., Moe, N.B., Vedal, H., Berntzen, M.: Using objectives and key results (OKRs) and slack: a case study of coordination in large-scale distributed agile. In: *Proceedings of the 55th Hawaii International Conference on System Sciences* (2021)
19. Uludağ, Ö., Philipp, P., Putta, A., Paasivaara, M., Lassenius, C., Matthes, F.: Revealing the state of the art of large-scale agile development research: a systematic mapping study. *J. Syst. Softw.* **194**(3), 212–220 (2022)
20. Yin, R.K.: *Case Study Research and Applications*. Sage, Thousand Oaks (2018)

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.



AI-assisted Agile



ChatGPT as a Tool for User Story Quality Evaluation: Trustworthy Out of the Box?

Krishna Ronanki^(✉), Beatriz Cabrero-Daniel, and Christian Berger

University of Gothenburg, Gothenburg, Sweden
{krishna.ronanki,beatriz.cabrero-daniel,christian.berger}@gu.se

Abstract. In Agile software development, user stories play a vital role in capturing and conveying end-user needs, prioritizing features, and facilitating communication and collaboration within development teams. However, automated methods for evaluating user stories require training in NLP tools and can be time-consuming to develop and integrate. This study explores using ChatGPT for user story quality evaluation and compares its performance with an existing benchmark. Our study shows that ChatGPT's evaluation aligns well with human evaluation, and we propose a “best of three” strategy to improve its output stability. We also discuss the concept of trustworthiness in AI and its implications for non-experts using ChatGPT's unprocessed outputs. Our research contributes to understanding the reliability and applicability of Generative AI in user story evaluation and offers recommendations for future research.

Keywords: ChatGPT · User Stories · Quality · Agile

1 Introduction

In agile software development projects, user stories are one of the most widely used notation to express requirements [1]. They are considered a very granular representation of requirements that developers use to build new features [2] as they help to capture & communicate end-user needs to prioritize & deliver small, working features in each development cycle [3].

The quality of user stories is crucial to the success of a development project as they impact the quality of the system design which, in turn, affects the final product [4]. They provide clear guidance for development efforts, improve communication and collaboration within teams, and help to ensure that development teams have a shared understanding of what needs to be delivered [5].

However, evaluating the quality of user stories manually can be time-consuming. One potential solution for improving agile software development processes is the integration of automated methods. This can be accomplished through modifications to existing workflows and the implementation of evaluation tools [6]. Existing methods for automatically evaluating user stories can be relatively fast and efficient, especially when compared to the time required for human evaluation [7]. Natural Language Processing (NLP) has been identified

as a potential method for evaluating various aspects of user stories. However, the accuracy and effectiveness of this method can be influenced by factors such as the quality of the training data and the complexity of the user stories under evaluation [8]. Unfortunately, the process of developing and incorporating automated methods for evaluating user stories can be a time-intensive endeavour due to the necessity of training NLP tools to accurately construct algorithms [9].

Developers are increasingly exploring the use of standalone general-purpose applications such as ChatGPT to aid in their software development endeavours. ChatGPT, based on the GPT-3.5 language model, is optimized for dialogue and is capable of answering questions in a human-like text [10].

Despite being trained on a large general-purpose corpus and specifically fine-tuned for conversational tasks [11], it has been observed to perform surprisingly well on specific technical tasks [12]. For this study, we investigated how well a general-purpose large language model like ChatGPT performs in evaluating the quality of user stories.

2 Background

The user story technique is a widely used approach for expressing requirements by utilizing a template that consists of the following elements: “As a (role), I want (goal), so that (benefit)” [3]. The primary components of a requirement that are captured by user stories are: who is it for, what it expects from the system, and, optionally, why it is important [3]. We follow this user story structure in our study while using the few-shot prompting technique to evaluate the user story quality using ChatGPT.

Few-shot prompting is a technique where the model is provided with a small number of examples of the task as conditioning in the initial prompt [13]. It refers to the ability of language models to learn a new task with limited training samples provided by the user [14, 15]. We used this prompting technique to provide an example to ChatGPT of what a user story should look like structurally before asking it to evaluate the user story on the defined criteria.

The user story quality criteria we used in our study were presented by Lucassen et al. [16] in their work which focuses on proposing a holistic approach for ensuring the quality of agile requirements expressed as user stories. The approach is comprised of two components: (i) the QUS framework, which is a collection of 13 criteria that can be applied to a set of user stories to assess the quality of individual stories and the set, and (ii) the AQUASA software tool, which utilizes state-of-the-art NLP techniques to automatically detect violations of a selection of the quality criteria in the QUS framework. Tõemets’ work investigates whether it is feasible to predict the quality of user stories for monitoring purposes and to determine the correlation between user story quality and other aspects of software development [17]. The user stories we chose to evaluate as part of our study and the benchmark evaluation scores of the selected user stories using the AQUASA tool were also included in this work.

3 Method

In our study, we performed a comparative analysis of manual and automated evaluation of user stories. Firstly, we assessed the quality of user stories manually, and then we employed ChatGPT for the same task. Our aim was to determine the effectiveness of ChatGPT in evaluating user stories and to compare its performance with human evaluation (Fig. 1).

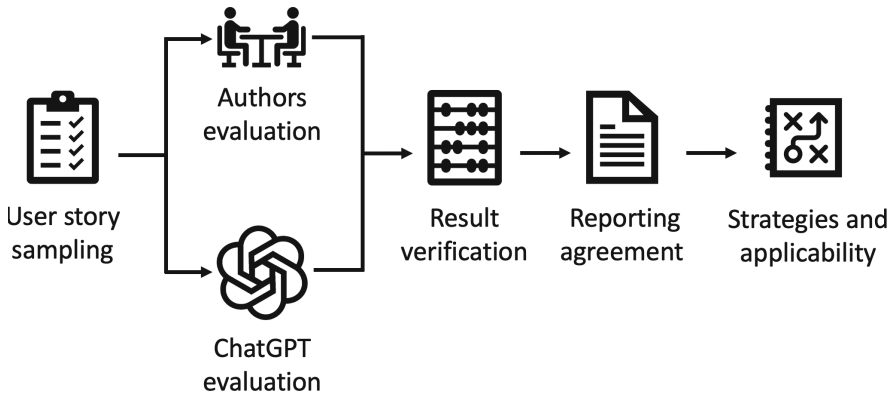


Fig. 1. Methodology and verification plan (during workshop)

To assess the ability of ChatGPT to replicate human evaluations of user stories, an open-source database presented in Töemets [17]¹ was selected as it came with benchmark evaluation of the user stories using the AQUSA tool, which also allows us to refer to an accepted baseline. After retrieving the benchmark, we conducted a double-blinded manual evaluation of the randomly selected set of user stories to assess their quality in terms of atomicity, well-formedness, minimality, conceptual soundness, unambiguity, completeness (full sentence or not), and estimability. The sole criterion for selection was the presence of a benchmark evaluation established using the AQUSA tool. However, the evaluation done by AQUSA presented in their study focuses on appraising only the following aspects: atomicity, well-formedness, and minimality.

To evaluate the performance of ChatGPT (March 23 version) for user story quality evaluation, we conducted a series of tests using the one-shot prompting method [15], a variation of the few-shot prompting method. Specifically, we presented ChatGPT with a set of criteria, user story pairs and recorded its responses. To ensure the reliability and consistency of ChatGPT's performance, we repeated this process three times. The evaluation was carried out based on seven criteria presented by Lucassen et al. [16].

¹ Visit <https://github.com/TanelToemets/Analysing-The-Quality-Of-User-Stories-In-Agile-Software-Projects>.

Finally, we compare the data from the human evaluation, the AQUUSA tool benchmark evaluations and the ChatGPT evaluation and present our findings as tables. The comparison was done for each of the seven criteria and for the overall precision, recall, specificity, and F1 score. The comparison was made to identify any significant differences between the two tools and to ascertain the accuracy of ChatGPT in replicating human evaluation.

The results of our experiments raise important issues related to the usability and transparency of ChatGPT's outputs, particularly for non-expert users. In this regard, the discussion section of our paper highlights the need to carefully consider the trustworthiness of ChatGPT's raw outputs and the importance of ensuring that users have the necessary tools to understand and interpret them correctly. By addressing these concerns, we can enhance the usability and effectiveness of ChatGPT as a tool for supporting decision-making in a variety of contexts.

3.1 Threats to Validity

Validity threats can arise in the benchmark creation process due to the limited scope of evaluation criteria used. The authors of the AQUUSA tool evaluated only the atomic, well-formed, and minimal criteria for the sampled user stories. Furthermore, there are concerns about the reliability and accuracy of the evaluation data since it was not provided by the AQUUSA authors themselves, but from a master thesis based on Lucasen et al. [16]. Another potential threat to validity is the use of human raters who may not have been experienced practitioners, thus leading to concerns about their reliability. To mitigate this, an independent rating of user stories was conducted, and in case of disagreement, a consensus was reached through a meeting. Moreover, ChatGPT was tested only three times, and further testing could yield different results. Nonetheless, we argue that this is sufficient to assert that developers cannot blindly trust ChatGPT's outputs and integrate them into their agile software development process. This finding emphasizes the need for cautious and careful consideration when incorporating natural language processing (NLP) tools like ChatGPT into agile development practices.

4 Results and Analysis

4.1 Comparing the Evaluations to the AQUUSA Benchmark

The AQUUSA benchmark comprises three key criteria for assessing the quality of a story. The first criterion is whether the story is well-formed, which means it includes a role and the expected functionality, commonly referred to as the means. The second criterion is whether the story is atomic, which implies that it addresses only one feature. The third criterion is whether the story is minimal, which requires that it contains a role, a means, and one or more ends [16].

Of all pairs {criteria, user story}, results showed that human evaluators and AQUUSA agreed on only **55%** of the pairs {criteria, user story} as reported in

Table 1, indicating a moderate level of agreement between the two methods. Human evaluators and AQUASA were in agreement in identifying well-formed and atomic user stories in a majority of cases (81.82% and 63.64%, respectively). However, the agreement rate between the two parties dropped significantly when it came to identifying minimal user stories, with only 18.18% agreement observed. The findings of the study indicate that the AQUASA tool exhibits a moderate level of concurrence with human evaluators when it comes to detecting user stories that are well-constructed and atomic in nature, but it currently falls short in identifying minimal user stories.

To enable a fair comparison of results, we conducted evaluations of the same user stories using ChatGPT. The evaluations were performed using two distinct accounts with the history log being cleared between each evaluation. We repeated this process thrice to account for any instability in the results. Table 1 displays the results of three evaluations conducted to assess the agreement rate and F1 scores of ChatGPT. The findings reveal that ChatGPT demonstrated a consistent agreement rate throughout the evaluations. Furthermore, the F1 scores recorded during the assessments ranged from 81% to 86%.

Table 1. Percentage of agreement, rounded to 2 decimals, between human evaluations and two tools, AQUASA and ChatGPT, across 11 randomly sampled user stories

Type	Metric	AQUASA	ChatGPT		
		Benchmark	Run 1	Run 2	Run 3
Criteria	Well-formed	81.82%	81.82%	81.82%	81.82%
	Atomic	63.64%	63.64%	90.91%	90.91%
	Minimal	18.18%	81.82%	72.73%	54.55%
Aggrega.	Agreement rate	54.55%	75.76%	81.82%	75.76%
	Precision	62.50%	80.95%	85.71%	74.07%
	Recall	71.43%	80.95%	85.71%	95.24%
	Specificity	25.00%	66.67%	75.00%	41.67%
	F1 score	66.67%	80.95%	85.71%	83.33%

4.2 ChatGPT-Human Agreement Rate

While performing the evaluation using ChatGPT, measures were taken to cover the rest of the metrics described in Dalpiaz et al. [16]. In terms of agreement rate with human evaluators, ChatGPT’s performance was relatively stable across the three assessments, as reported in Table 2, with agreement rates ranging from 73% to 75%. However, this suggests that there is room for improvement in the agreement rate between ChatGPT and human evaluators. A 25% error rate may be problematic in certain situations. As a result, enhancing ChatGPT’s performance could increase its reliability and effectiveness in various applications.

Table 2. Agreement with human evaluations of ChatGPT, across 11 randomly sampled user stories, using different strategies to interpret the output

Type	Metric	ChatGPT			Interpretation strategy		
		Run 1	Run 2	Run 3	AL1	BO3	PA3
Criteria	Well-formed	81.82%	81.82%	81.82%	81.82%	81.82%	88.89%
	Atomic	63.64%	90.91%	90.91%	72.73%	90.91%	100.00%
	Minimal	81.82%	72.73%	54.55%	45.45%	81.82%	100.00%
	Conceptually sound	90.91%	90.91%	63.64%	63.64%	81.82%	100.00%
	Unambiguous	54.55%	54.55%	63.64%	72.73%	54.55%	62.50%
	Full sentence	45.45%	63.64%	81.82%	81.82%	63.64%	80.00%
	Estimable	90.91%	72.73%	81.82%	72.73%	81.82%	88.89%
Aggrega.	Agreement rate	72.73%	75.32%	74.03%	70.13%	76.62%	87.23%
	Precision	83.33%	82.69%	77.05%	72.06%	81.82%	94.74%
	Recall	75.47%	81.13%	88.68%	92.45%	84.91%	90.00%
	Specificity	66.67%	62.50%	41.67%	20.83%	58.33%	71.43%
	F1 score	79.21%	81.90%	82.46%	80.99%	83.33%	92.31%
	Coverage	100.00%	100.00%	100.00%	100.00%	100.00%	61.04%

The agreement rates reported in Tables 1 and 2 include both true positives, where human raters and tools agreed on an overall positive evaluation of a user story, and true negatives, where humans and the tools agreed on a negative evaluation. Future work, though, might look into database entries where human raters and ChatGPT do not agree on the evaluations.

4.3 How to Select an Answer Based on ChatGPT’s Output

In our study, we evaluated the consistency and reliability of ChatGPT in evaluating user stories against a set of predetermined criteria. Our results indicate that ChatGPT was consistent with itself in 61% of the evaluations, meaning it gave the same response for a given pair {criteria, user story} in three separate runs (PA3). Furthermore, ChatGPT agreed with itself in at least two out of three runs in 83.11% of the evaluations. These findings suggest that ChatGPT’s evaluations are relatively stable when it comes to evaluating user stories. Moreover, we observed that in the subset of evaluations where ChatGPT was consistent across all three runs, the agreement rate with humans was 87%, with precision and recall scores of 95% and 90%, respectively.

The higher rate of agreement between humans and ChatGPT has encouraged us to explore stricter criteria for identifying positive responses. The use of a “best of three” approach in which ChatGPT is required to give a positive response in at least two out of three attempts resulted in a slightly higher agreement rate between humans and ChatGPT, reaching (77%). However, when responses from ChatGPT were classified as positive if at least one positive response was given (AL1), there was more variability in ChatGPT’s responses, leading to a decrease in the agreement rate to 70%.

5 Discussion

The agreement rate remained constant across the first three rounds, as evidenced by Tables 1 and 2. Thus, we opted to conclude our testing after these three runs. However, to establish the reliability of these initial findings, additional evaluations of new user stories and more ChatGPT runs are necessary.

Table 2 shows that the criteria with the highest and lowest agreement rates differed in the three runs, which suggests that certain criteria may have unclear definitions or abstract qualities that made it difficult for ChatGPT to consistently agree with human evaluators. To better understand this discrepancy, further research could examine the specific criteria that posed challenges for ChatGPT or where it exhibited inconsistencies.

Although ChatGPT's consistency in generating responses may correlate with the level of agreement from human evaluators, it is important to note that consistency does not necessarily equate to the accuracy or appropriateness of the output. Additionally, the consistency could be due to potential bias present in the training data.

However, integrating ChatGPT into agile software development requires a thorough assessment of its capabilities, strengths, and limitations. While ChatGPT has shown promise in this task, its performance is not flawless, and it remains an emerging technology that is susceptible to potential biases and limitations. Therefore, careful consideration of ChatGPT's applicability and limitations is necessary before its integration into agile software development processes [18]. On the other hand, GPT-4's expanded architectural model size might play a pivotal role in enhancing its proficiency in NLP, which could lead to increased accuracy and relevance in the generated responses [19].

However, a major obstacle to using ChatGPT in the requirements elicitation process is the issue of extrinsic hallucinations [20]. Non-experts who rely on AI systems might not possess the technical knowledge to evaluate the accuracy and reliability of the generated outputs in some cases. This issue highlights the importance of ensuring the trustworthiness of the AI systems being implemented. Ensuring trustworthiness in AI, particularly in the context of non-experts using ChatGPT for user story evaluation, requires careful consideration of several factors including transparency, explainability, bias mitigation, and continuous improvement through user feedback to be incorporated into the development and implemented process of these AI systems in such human-centric processes.

6 Conclusion and Future Work

The study examines the effectiveness of ChatGPT in assessing the quality of user stories, particularly when it produces consistent results across multiple evaluations. The research focuses on the agreement rate between humans and ChatGPT in evaluating user stories based on said criteria. The results indicate that ChatGPT is more capable of replicating human evaluation (approximately 75%) than the AQUASA tool as demonstrated in Tõemets [17].

While the model performs sufficiently well in independent runs, it exhibits inconsistency in its Boolean outputs when tested multiple times. This suggests caution in interpreting its evaluations and underlines the need for further research into the factors affecting ChatGPT's consistency and reliability.

To address the issue of unstable outputs, the paper suggests strategies such as selecting the “best of three” approach. However, the question of whether ChatGPT's raw outputs can be used directly by non-expert users raises important concerns about the trustworthiness of AI systems. As a result, high-level trustworthiness requirements must be established to ensure that ChatGPT and other AI tools are integrated into Agile software development processes following trustworthy AI principles. The integration of ChatGPT, into agile software development processes, requires careful consideration of its limitations and strengths and the potential impact on the development process. Further research is needed to explore ways to ensure that ChatGPT and other AI systems can be used reliably and effectively in Agile development environments.

References

1. Lucassen, G., Dalpiaz, F., Van Der Werf, J.M.E., Brinkkemper, S.: Forging high-quality user stories: towards a discipline for agile requirements. In: IEEE International Requirements Engineering Conference (RE), pp. 126–135. IEEE (2015)
2. Lucassen, G., Dalpiaz, F., Werf, J.M.E.M., Brinkkemper, S.: The use and effectiveness of user stories in practice. In: Daneva, M., Pastor, O. (eds.) REFSQ 2016. LNCS, vol. 9619, pp. 205–222. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-30282-9_14
3. Cohn, M.: *User Stories Applied: For Agile Software Development*. Addison-Wesley Professional, Boston (2004)
4. Amna, A.R., Poels, G.: Systematic literature mapping of user story research. *IEEE Access* **10**, 51723–51746 (2022)
5. Mustaffa, S.N.F.N.B., Sallim, J.B., Mohamed, R.B.: Enhancing high-quality user stories with AQUASA: an overview study of data cleaning process. In: 2021 International Conference on Software Engineering & Computer Systems and 4th International Conference on Computational Science and Information Management (ICSECS-ICOCSIM), pp. 295–300 (2021)
6. Humayoun, S.R., Dubinsky, Y., Catarci, T.: User evaluation support through development environment for agile software teams. In: Caporarello, L., Di Martino, B., Martinez, M. (eds.) *Smart Organizations and Smart Artifacts*. LNISO, vol. 7, pp. 183–191. Springer, Cham (2014). https://doi.org/10.1007/978-3-319-07040-7_18
7. Jurisch, M., Lusky, M., Iglar, B., Böhm, S.: Evaluating a recommendation system for user stories in mobile enterprise application development. *Int. J. Adv. Intell. Syst.* (2017)
8. Peña, F.J., Roldán, L., Vegetti, M.: User stories identification in software's issues records using natural language processing. In: 2020 IEEE Congreso Bional de Argentina (ARGENCON), pp. 1–7 (2020)
9. Sharir, O., Peleg, B., Shoham, Y.: The cost of training NLP models: a concise overview. [arXiv:2004.08900](https://arxiv.org/abs/2004.08900) (2020)
10. Zhang, B., Ding, D., Jing, L.: How would stance detection techniques evolve after the launch of ChatGPT? *arXiv preprint: arXiv:2212.14548* (2022)

11. Shen, Y., et al.: ChatGPT and other large language models are double-edged swords (2023)
12. Choi, J.H., Hickman, K.E., Monahan, A., Schwarcz, D.: ChatGPT goes to law school. Available at SSRN (2023)
13. Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., Sutskever, I., et al.: Language models are unsupervised multitask learners. *OpenAI Blog* **1**(8), 9 (2019)
14. Perez, E., Kiela, D., Cho, K.: True few-shot learning with language models. In: *Advances in Neural Information Processing Systems*, vol. 34, pp. 11054–11070 (2021)
15. Liu, P., Yuan, W., Fu, J., Jiang, Z., Hayashi, H., Neubig, G.: Pre-train, prompt, and predict: a systematic survey of prompting methods in natural language processing. *ACM Comput. Surv.* **55**(9), 1–35 (2023)
16. Lucassen, G., Dalpiaz, F., van der Werf, J.M.E., Brinkkemper, S.: improving agile requirements: the quality user story framework and tool. *Requirements Eng.* **21**, 383–403 (2015)
17. Tõemets, T.: Analysing the quality of user stories in open source projects. PhD thesis, University of Tartu (2020)
18. Borji, A.: A categorical archive of ChatGPT failures (2023)
19. Koubaa, A.: GPT-4 vs. GPT-3.5: a concise showdown. Available in <https://doi.org/10.36227/techrxiv.22312330> (2023)
20. Bang, Y., et al.: A multitask, multilingual, multimodal evaluation of ChatGPT on reasoning, hallucination, and interactivity. arXiv preprint: [arXiv:2302.04023](https://arxiv.org/abs/2302.04023) (2023)

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.





Survey of AI Tool Usage in Programming Course: Early Observations

Mika Saari^(✉), Petri Rantanen, Mikko Nurminen, Terhi Kilamo, Kari Systä,
and Pekka Abrahamsson

Tampere University, Tampere, Finland
`mika.saari@tuni.fi`

Abstract. This study explores the role of artificial intelligence (AI) in university teaching, with a focus on the teaching of programming. Despite the growing use of AI in education, both students and teachers often struggle to understand its role and implications. To address this gap, we conducted a survey of a wide group of students ($n = 200$) to assess their experiences and perspectives on the use of AI in programming education. The survey results suggest that AI is becoming increasingly involved in university teaching, particularly in the teaching of programming. However, students require guidance from teachers in order to use AI tools effectively in their studies. The study concludes that the goal of teaching programming should be to guide students in the responsible use of AI. Overall, the study highlights the need for greater awareness and understanding of AI in university teaching and the fact that teachers have an important role to play in providing guidance to students on the responsible use of AI tools.

Keywords: AI · Education · Programming

1 Introduction

AI assisted learning, e.g., the use of conversational style language models like ChatGPT¹, has the potential to revolutionize the way programming courses are taught. With artificial intelligence (AI), students can receive tailored, personalized and interactive support, receive immediate feedback, and have access to a virtual tutor 24/7. AI can also analyze student's performance and provide suggestions for improvement, making the learning experience more efficient and effective. Additionally, AI can automate tedious tasks such as grading code, freeing up instructors to focus on more important tasks such as providing meaningful feedback and engaging in interactive discussions.

The study by Adamson [1] considers the usability of artificial intelligence through different perspectives. They discuss the problems of AI systems because they have black box characteristics and therefore some checks that are consistent

¹ <https://chat.openai.com>.

with philosophical scrutiny are suggested. Finally, they take an example of recent experimentation with ChatGPT and also generally highlight the challenges and expectations of AI systems.

Use of AI in university education can be seen from two different perspectives: the student's perspective and the teacher's. This study focuses on the use of AI from the students' point of view. The study aims to investigate how much students already use AI to support their studies and for what kind of purposes AI tools have been used. This can help teachers to consider the pedagogical viewpoint and support the selection of pedagogical approaches that enable learning of the right things with AI tools. Thus, the research question was formed:

RQ: How widely have students adopted AI tools in their studies and what are their motives for and against the use of AI?

To answer the research question, a survey was conducted among students enrolled in a BSc level programming course to evaluate the use of AI as a tool for supporting their study of the subject.

The rest of the paper is structured as follows: Sect. 2 clarifies the research environment with teaching, including recent AI related studies. Section 3 focuses on the research aims, context, and data collection and analysis methods. Section 4 contains results with analysis and discusses the findings. Finally, Sect. 6 summarizes the research.

2 Background

According to an international survey of over 500 students and teachers, difficulties in learning programming are common, particularly at the basic level. The survey results were analyzed to identify specific challenges and to make recommendations for improving programming education. According to the survey, students consider practical programming exercises that require actively applying participation to be the most useful for learning purposes [2].

Another study [3] reported that self-organized learning is increasingly supported by personal learning environments (PLE), where the learner is in control of their own development and learning. The PLE concept translates the principles of self-organized learning into actual practice. The PLE-driven approach puts the learner at the center and gives them control over the learning experience. In addition, PLEs aim to provide learners with tools and resources that support continuous learning. The PLE approach recognizes that learning is a lifelong process and seeks to enable learners to take control of their own learning by providing them with the necessary tools and resources [4].

Additionally, question and answer sites, such as Stack Overflow, are often used for finding help on programming related tasks. How students use Stack Overflow was studied in detail by [5]. Traditionally, search engines have been used to seek further assistance for programming tasks. However, a new trend is emerging where people are turning to AI-based solutions. These solutions may take the form of virtual assistants or chatbots, such as Siri, Alexa, or Google

Assistant [6]. The increasing popularity of AI-assisted learning, utilizing chatbots as a means of providing personalized and interactive educational experiences, can be seen by taking a look at the recent publications in the IEEE Xplore database. For example, using the keywords “chatbot” and “education” on April 12th, 2023, yielded 251 results, 79% (198) of which were published in the last three years.

In the context of this research, the keyword “programming course” was used to identify relevant studies, resulting in a total of 16 publications. Of these, three studies in particular, namely [7,8], and [9], emphasized the role of instructional guidance during the learning process.

The focus of the study by Verleger [7] was on the creation of an intelligent chatbot interface designed for an introductory computer programming course. This chatbot, named “EduBot”, was initially equipped with a limited knowledge base, as the intention was to expand it gradually based on the interactions and feedback from the students. If the question was new to the chatbot, the instructor would answer and the database was expanded. The chatbot was based on data from the Canvas Learning Management System² and Microsoft QnA Maker³ [7].

Another study [8] introduced an AI chatbot to help students with academic issues. The chatbot was based on IBM Watson⁴ and a predefined question bank. According to the study, students found the tool useful for introductory programming [8].

A study by Carreira [9] used a chatbot (Pyo) using Rasa⁵ and the Python programming language. The Pyo chatbot has three primary functions: exercise assistance, error guidance, and concept definitions. The chatbot was seen as useful for students in their programming studies because it offers benefits such as immediate response and feedback, and individualized support [9].

3 Research Approach

The research aims, context, data collection, and analysis methods are presented in this section. In this study, the content analysis method was used to evaluate the answers received in the survey.

The aim of this study was to investigate students’ use of artificial intelligence in their programming studies. Tampere University published “Tampere University’s guidelines on the use of AI applications” at the end of January to clarify the rules of usage. The main idea of the guidelines is that the usage of AI applications is allowed, but if students use them, they have to mention it. The usage of AI tools in programming courses was not specifically mentioned.

In the study, the research aim was to clarify the current usage of AI tools when studying Java programming. At first, we wanted to find out how the publicity received by artificial intelligence had affected students’ behavior when studying

² <https://www.instructure.com/canvas>.

³ <https://www.qnamaker.ai/>.

⁴ www.ibm.com/watson.

⁵ rasa.com.

programming. Before this study, we assumed that students might use two different AIs: the ChatGPT, which was launched in November 2022, and the Github Copilot, which was released in June 2021.

With these initial assumptions, the objective of our study was to investigate the opinions and attitudes of respondents toward AI in learning. In addition to this, we aimed to identify new and emerging challenges as well as the benefits of using AI by analyzing the survey findings. We also wanted to focus on detecting any significant differences in the underlying perceptions between respondent groups with a negative or positive attitude toward AI adoption. By achieving these goals, we hope to gain a deeper understanding of the current perceptions and attitudes toward AI and provide valuable insights that can be used to improve the adoption of this technology in education.

The **research context** was the programming course offered to Bachelor's and first year Master's level students who had already completed at least two previous courses dealing with the basics of programming. The questionnaire was arranged as part of the "Programming 3: Interfaces and Techniques (5 cr)" course at Tampere University. The course started in January and ended in May. The course had about 360 registered students from different starting years (Table 1). The data are from the Tampere University student information database, but it is notable that these data do not include knowledge about gender. Typically, students complete the course in the spring of their second year of study, but depending on the degree program there is a slight variation due to scheduling of the programming courses. Based on previous implementations of the "Programming 3" course, it is estimated that about 60–70% of those who register will complete the course. The course includes several theory and programming exercises for which students collect points. The questionnaire was defined as a theory exercise that earned points, but the quality of the answers was not evaluated. Therefore, the response percentage of 58 is an average response level for theory exercises.

The data for this study was collected through a survey administered to students enrolled in the "Programming 3" course. The survey was conducted halfway through the course. An online survey was organized based on the research questions. The survey was offered to the students using the same Plussa Learning Management System (LMS) as for returning other exercises. Plussa is based on Aalto A+ LMS [10], which was developed by Aalto University.

It should be noted that the survey was not directly related to the students' programming studies. Two questions were used to conduct the survey, which were designed to elicit detailed responses and provide valuable information for research.

The survey was available in both Finnish and English. The first part (Q1) aimed to determine the extent of artificial intelligence usage, while the second part (Q2) aimed to investigate the manner in which students utilized artificial intelligence.

Q1: Have you used AI (e.g., ChatGPT) in your studies?

Q2: In what situations and why have you used AI? If you have not, please elaborate on your reasons for not using AI.

The first question (Q1) was formulated as a multiple-choice question and the possible choices were never, once or twice, occasionally, weekly, and daily. The second (Q2) was an open-ended question that allowed students to provide a detailed and personalized answer in their own words.

4 Results of the Survey

In this section, we start by answering the research question. The first multiple-choice question asked how extensively students used AI, whereas the second open-ended question aimed to understand their motivation for using or not using AI. The survey results are presented and the findings discussed in relation to teaching programming. At this point in the course, artificial intelligence tools had been briefly explained and discussed, but their use to support learning had not been covered in any way.

Multiple-choice question: Have you used AI (for example ChatGPT) in your studies?

About one third of the students had never used AI tools, such as ChatGPT, in their studies. However, a significant proportion of students had used AI tools once or twice (26%) or occasionally (19%). Only a minority of students reported using AI tools more frequently, with 17% using them on a weekly basis and 3% using them daily.

Open-ended question: In what situations and why have you used AI? If you have not, please elaborate on your reasons for not using AI.

With this question we wanted to resolve whether the respondents had utilized AI in any circumstances and if so, what were the reasons for doing so? If they had not made use of AI, we wanted them to explain the rationale behind that decision.

We received around 200 verbal answers to the open-ended question, with lengths varying from 5 to 200 words. These student answers were collected and stored in an answer database. The following steps were taken to evaluate the answers:

1. Data about the starting year were added to the answer database.
2. The answers were divided into three categories: AI Used, AI Not Used, and AI Tried. The results of steps 1 and 2 are shown in Table 1.
3. First, the answers were evaluated by reading them through and looking for keywords. At this stage, the keywords that appeared most frequently in the answers were selected. Finally, keywords such as “Speed up learning”, “Clarifying things”, “Code syntax clarification”, and “Learning in general” were combined into the category “Learning”. A similar process was used to define the categories “Search engine”, “Coding,” and “Debugging” as shown in Fig. 1 (“Used” answers).
4. Evaluating and categorizing of the “Not Used” answers gave the result shown in Fig. 2.

- Evaluating the “Tried” answers. There was no clear division of reasons. Based on the answers, it is worth noting that if problems are encountered in the use of AI at the beginning, students may stop using it.

Table 1. Student Enrollment by Starting Year. The “Used” statistics are calculated from the open-ended Q2.

Starting year	Used	Not Used	Tried	Participants
2022	72%	19%	8%	36
2021	59%	31%	10%	93
2020	56%	42%	3%	36
2019	53%	32%	16%	19
2018 and earlier	62%	38%	0%	13
All	60%	32%	8%	197

According to Table 1, 60% of participants reported using AI during their studies, and the proportion was larger among students who had started their studies in 2022. It should be noted that the students who started in 2022 took the course significantly earlier than recommended. Thus, they may represent a special group.

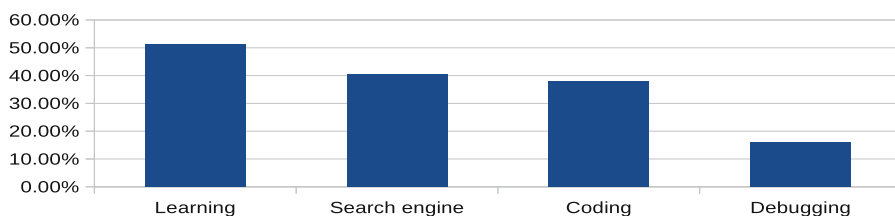


Fig. 1. Categories of “Used” answers based on reasons to use AI during studies.

Figure 1 provides information regarding the use of AI in the context of studying. The values regarding the use of AI were formed through the evaluation of verbal responses, performing searches by using key phrases (such as “to speed up learning”) to enable later categorizing of answers. The term “learning” includes aspects like speeding up learning, clarifying things, and clarifying code syntax. Several students compared the ChatGPT tool to Google and reported the faster speed of information retrieval with ChatGPT. In the “coding” option, the coding of individual parts of the assignments was frequently mentioned, but some students also reported using AI comprehensively to solve practical exercises. The term “debugging” came up frequently when trying to determine the meaning of error messages given by the program.

It is worth noting that one answer could include several of the key phrases, so the bars of Fig. 1 are compared individually to the total answers.

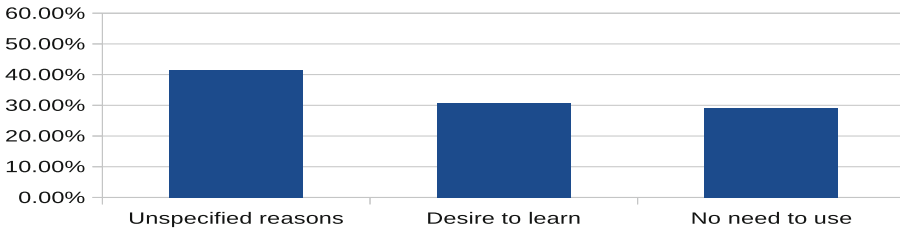


Fig. 2. Categorize of “Not Used” answers based on reasons.

Figure 2 provides information regarding why students did not use AI in the context of studying. Unspecified reasons mostly include answers like “I didn’t feel like doing it” or “I didn’t bother to do it.” The reason “desire to learn” mostly described situations where the student’s opinion reflected the excessive ease of AI when doing the exercises and therefore they had a negative attitude toward using AI. The remainder of the students thought that they did not need to use AI, because they could handle the exercises without it.

The third category, “AI Tried,” has a clear division of aspects. The most common answer was similar to “I have tried, but I did not get any benefit from it” or “it did not work as well as I expected”.

5 Discussion

Analyzing the results of the students’ self-initiated use of artificial intelligence during their studies was carried out to answer the research question. Additionally, their motivation for and against AI usage were also taken into consideration during the analysis.

In this study, the results were based on the collected written comments from students, which were categorized according to the previously defined categories. Additional depth can be added to this analysis by featuring a few typical “AI used” comments provided by the students.

Several responses were categorized under “learning”. For instance, the comment “I use it to: - quiz me about exam topics - making the text material more clear and also asking questions about the text - if my code has a bug i ask about it” is a great example of the educational application of AI tools. The student uses the AI tool as an instructor, facilitating understanding in various ways.

One such common response when asked about the utility of ChatGPT as a search tool is, “AI is pretty useful for doing templates and getting the starting point on some questions. On the other and if you have a specific problem AI is faster in solving your problem then searching the internet yourself.” This response highlights the value and robust knowledge that AI tools can provide.

Students also utilize AI tools for coding and debugging purposes. For instance, the response “I used AI when I need more information about some concepts or how to handle data in Java” highlights the capability of AI in explaining unfamiliar programming language techniques. Conversely, the comment “I usually use AI to fix my own program or more likely to debug what is wrong with my program. But I also have tried once or twice using the AI to do a task that I don’t fully understand” falls under both the “Coding” and “Debugging” categories. This comment indicates that some students use AI tools for coding even without complete understanding. Such an approach is not ideal and students should be guided to avoid it.

Furthermore, an analysis of the “tried” responses revealed that these answers were generally longer, and the reasons provided were more diverse. Common phrases such as “It is too easy” or “I want to learn by myself” were frequently mentioned. These responses indicate that students perceive AI tools as being too powerful to use. In the future, providing instruction and guidance on how to use AI tools could help address these concerns.

The comments like “I didn’t find it useful” shows that not all students find AI tools worth their time. Though it is difficult to say whether they didn’t know how to use the tools or they simply didn’t find the tools useful. The statement “It did not work as I supposed” suggests more clearly that students had technical difficulties. Overall, when students encounter difficulties with new technology, it may be easier to set the tools aside and not utilize them. There were many answers with comments similar to either to those presented here, but students’ answers did not indicate one clear reason for the problems they encountered.

According to the results, the majority of students use AI in their studies. The results also indicate that it cannot be concluded that the students are generally against the use of artificial intelligence.

One potential problem with Q1 was discovered during the analysis phase. Students were presented with the question: “Have you used AI (e.g., ChatGPT) in your studies?” The explicit mention of ChatGPT might have directed students to consider only chat-based AI interactions, causing them to base their answers solely on using chat-based AI or even only ChatGPT itself. However, apart from chat-based AI tools, students might have also utilized other AI tools mentioned previously, such as GitHub Copilot, or other AI tools that provide code generation assistance based on students’ input, including comment lines. Additionally, students likely employed code completion tools like IntelliSense. The ubiquity of these types of AI systems in assisting coding tasks is so common that students might not consider them at all when discussing their use of AI tools.

However, from the survey results, it can be inferred that some do not want to use AI in their studies because they consider it too easy a way to complete a course. Surprisingly, some students believe that the use of AI is prohibited. Nevertheless, the university has informed students that it allows the use of AI and has also provided guidelines for its use in studying. This shows a clear need for developing guidelines for the use of AI at the university. These issues pose a challenge for teachers when considering the use of AI in developing study materials and assignments.

The results of this study should be factored in when designing material on AI usage in the programming course. We also aim to repeat this questionnaire when the course is next held. This will also provide the opportunity to compare the changes in opinions toward AI.

Further research could explore the reasons behind the variations in usage frequency and identify potential opportunities for promoting AI tool adoption among students.

6 Summary

The results of this research suggest that the use of AI is becoming more prevalent in university teaching, particularly in the teaching of programming. However, both students and teachers have difficulty understanding the role of AI in education. The research conducted a survey of a wide group of students. A study showed that students widely use AI tools as learning aids, search engines, and coding assistants. The main conclusions drawn from the results is that the goal of teaching programming should be to guide students in the responsible use of AI in their studies. This includes teaching them about AI tools and how to use them effectively, as previously mentioned, for learning, knowledge searching, and coding assistance. Teachers have an important role to play in providing this guidance. Overall, the research highlights the need for greater awareness and understanding of AI in university teaching.

References

1. Adamson, G.: Explaining technology we don't understand. *IEEE Trans. Technol. Soc.* **4**(1), 34–45 (2023)
2. Lahtinen, E., Ala-Mutka, K., Järvinen, H.-M.: A study of the difficulties of novice programmers. *ACM SIGCSE Bull.* **37**, 14–18 (2005)
3. Chatti, M.A., Agustiawan, M.R., Jarke, M., Specht, M.: Toward a personal learning environment framework. *Int. J. Virtual Pers. Learn. Environ.* **1**, 66–85 (2010)
4. Attwell, G., et al.: Personal learning environments - the future of elearning. *Elearning Papers* **2**(1), 1–8 (2007)
5. Robinson, D.: How Do Students Use Stack Overflow? (2017). Accessed 29 Mar 2023
6. Luger, E., Sellen, A.: Like having a really bad PA, pp. 5286–5297. *ACM*, May 2016
7. Verleger, M., Pembridge, J.: A pilot study integrating an AI-driven chatbot in an introductory programming course, pp. 1–4. *IEEE*, October 2018
8. Ismail, M., Ade-Ibijola, A.: Lecturer's apprentice: a chatbot for assisting novice programmers, pp. 1–8. *IEEE*, November 2019
9. Carreira, G., Silva, L., Mendes, A.J., Oliveira, H.G.: PYO, a chatbot assistant for introductory programming students, pp. 1–6. *IEEE*, November 2022
10. Karavirta, V., Ihanola, P., Koskinen, T.: Service-oriented approach to improve interoperability of e-learning systems. In: 2013 *IEEE 13th International Conference on Advanced Learning Technologies* (2013)





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.





Turning Large Language Models into AI Assistants for Startups Using Prompt Patterns

Xiaofeng Wang^(✉), Mohammad Idris Attal, Usman Rafiq,
and Sylvia Hubner-Benz

Free University of Bozen-Bolzano, Bolzano 39100, Italy
{xiaofeng.wang,mohammadidris.attal,urafiq,sylvia.hubner}@unibz.it

Abstract. Most startups operate with limited resources and experience. AI technologies enable them to accomplish many tasks under these constraints. The recent advance of large language models (LLMs) offers new opportunities to support startup endeavors. Given the nascent nature of LLMs, how they could be utilized to support startups is yet to be investigated. Since prompt engineering is believed to be at the core of the effective use of LLMs, we aim to understand how to apply prompt engineering to turn LLMs into AI assistants for startups. As the first step, we investigated the application of a set of prompt patterns to ChatGPT, arguably the most widely known LLM currently. The preliminary results show that some patterns are more suitable for brainstorming which is a typical activity conducted by early-stage startups. Prompt-tuned questions may lead to more specific and more detailed responses, but it is not guaranteed. Meantime, human factors play an important role in the effective application of prompt patterns. Large-size and systematic studies are needed to apply the right patterns to different questions, taking into account the differences among startups in terms of their startup knowledge, domain knowledge, and their attitudes and behaviors towards LLMs.

Keywords: Startups · AI Assistant · Large Language Models · Prompt Engineering · Prompt Pattern

1 Introduction

Building a startup is a challenging endeavor, and the failure rate is notoriously high¹. Scarce resources and lack of knowledge and experience in startup processes are among the key reasons why startups fail. Support for startups exists in different forms, such as mentoring, incubation, and digital tools. With the emergence of Generative Artificial Intelligence (GAI), particularly Large Language Models (LLMs), startups now have more opportunities to receive assistance and develop their innovative ideas. However, given the nascent nature of LLMs, how they could be utilized to support startups is yet to be investigated.

The emergent and enduring value of LLMs is fundamentally tied to effective prompt engineering [1]. A *prompt* refers to a set of text instructions crafted to

¹ <https://explodingtopics.com/blog/startup-failure-stats>.

program and customize LLMs for the desired interaction. It provides a scope or context for an LLM to act on [2]. In turn, *prompt engineering* is the means by which LLMs are programmed via prompts [3]. Prompt engineering skills are vital for fully leveraging LLMs, but they do not come naturally and need to be learned. This can pose a challenge to startup teams, who often operate on tight resources and have many other critical tasks to attend to [4]. We aim to facilitate startups in utilizing LLMs as AI assistants through prompt engineering. To this end, we propose the research question: *How to apply prompt engineering to turn large language models into AI assistants for startups?*

As the first step to answer the research question, we investigated prompt patterns which are summaries of effective prompt-tuning techniques. They provide a codified approach to customizing the input and output of LLMs as well as interactions with them. Application of prompt patterns may prove beneficial for startups in maximizing the potential of LLMs. We evaluated the prompt patterns proposed in [3], and identified a subset of them as more relevant in the startup context (see Sect. 2). We first tried these patterns using a simulated conversation with ChatGPT. Then we applied them in real-life conversations between a group of students studying entrepreneurship and ChatGPT. The preliminary results from this initial step helped us achieve a better understanding of the requirements for converting LLMs into effective AI assistants.

2 Background and Related Literature

The role of AI in transforming businesses is now well established. However, its utility in the context of startups has remained fuzzy [5]. While several startups are seen using this technology to support their operations in data analysis, chatbots, and process automation, a vast majority of startups are still unsure how to incorporate or leverage this technology in the development of their innovative ideas. The challenge becomes more significant when considering early-phase startup development in which ideation and brainstorming activities are intensive. The existing literature paid little attention to the role of AI and how to leverage its core values for startups [5]. The recent upsurge of generative AI, particularly LLMs, calls for more exploration of AI potentials for startups.

The development of the first GPT (Generative Pre-trained Transformer) model in 2018 [6] laid the foundation of LLMs. They gained significant attention soon after OpenAI released ChatGPT on November 30, 2022. Dimitrov [7] defines an LLM as a pre-trained model, trained on a very large text dataset. LLMs promise great potential in generating, summarizing, translating, and performing various natural language tasks [8]. They can offer a series of conversations with a great conversational user experience [9]. However, despite the widespread adoption of LLMs in various industries, it is unclear how non-experts can design effective prompts to interact with these models and elicit the desired behaviour [9].

Dwivedi et al. [10] claim that the output of an LLM significantly depends on how a prompt is designed and provided to the model. Pengfei et al. [2] conclude the same and propose to experiment with prompts to elicit the desired knowledge from LLMs. A similar view is presented in another study [7]. The author ascertains

that a response from an LLM will be effective if a prompt is good. Therefore, effective prompts are essential for a desired answer and future interactions. Similarly, according to [10], there is a need to train users to provide an effective prompt and it is going to be an essential future skill. The field to design and implement prompts for LLMs is referred to as prompt engineering [3].

In this context, White et al. [3] propose a catalog of prompt patterns to enhance the output of LLMs. The authors propose 15 prompt patterns in their study, originally inspired by the software design patterns. The presented patterns are designed with the intent to offer better yet reusable ways to interact with LLMs. Considering all the patterns in the catalog, we selected seven patterns to be further investigated in our study, briefly described below.

- **Persona:** In this pattern, a user asks an LLM to play a particular role. In role-playing, the LLM is given a role without providing fine details of that role. The pattern is applicable when users do not know the exact details required to process the request, however, what they know is the role of a person who is responsible for this kind of job.
- **Context Manager:** This pattern helps users to either introduce or remove a specific context while having a conversation with LLM. Therefore, users drive LLMs to consider or ignore a few aspects while producing the output. Otherwise, LLMs tend to provide broader or generic answers to a particular question asked by the user.
- **Flipped Interaction:** In this pattern, the interaction between the user and LLM is flipped. It means that the LLM is supposed to lead the interaction and ask questions to accomplish the user’s goals. Communicating a goal to the LLM is a prerequisite in this pattern. As the content of the interaction is produced by the LLM according to the specified goal, therefore, a more precise output is generated by the LLM using the knowledge that the user does not possess.
- **Cognitive Verifier:** This pattern is proposed to restrict LLMs to always decompose the original questions into a series of sub-questions automatically. Thereafter, by combining answers to sub-questions, an LLM produces the answer to the original question. The original thought for this pattern comes from a recent study [11]. According to the authors, the LLM can assert more reasonably if we divide the key problem into sub-problems and then the LLM can process them in a sequence. This strategy is referred to as least-to-most prompting [11]. The primary goal of this pattern is to improve the prompts.
- **Alternative Approaches:** It helps to overcome the problem of cognitive biases. Humans naturally have the tendency to exhibit what they see and think. Therefore, the rationale behind this pattern is to overcome cognitive errors so that users may ask for alternative ways of doing a particular task. A comparison of alternative practices, in terms of pros and cons, could also be asked by users.
- **Question Refinement:** The purpose of this pattern is to prompt an LLM to produce a refined version of the user question so that a piece of accurate information should be produced. Therefore, the LLM needs to have a few

interactions with a user to produce the refined question. Alongside this, the LLM also needs a context to produce a better version of the question. Therefore, the goal is to turn the user's question into a refined version with the help of a series of interactions.

- **Template:** This pattern is recommended when the user needs to restrict LLMs to follow a use-case and produce output accordingly. Therefore, LLMs deliver output in a format that the user specifies. In this scenario, LLMs do not have knowledge of the specified template and the user has to specify it while asking a question.

3 Research Process

We conducted the study in two steps: 1) Apply the prompt patterns to a simulated conversation, and 2) Apply the prompt patterns to real-life conversations.

In the first step, we created a scenario in which a younger entrepreneur is interested in building a startup in the healthcare domain. She did not have a clear startup idea to start with and turned to ChatGPT to understand how she could proceed. We designed the conversation following an example provided by an entrepreneurship educator in a YouTube video². The educator is an expert on entrepreneurship education³. The conversation covers various aspects of the ideation phase, such as problem and solution, customer segment, first minimal viable product, etc. We created two versions of the conversation, the first one with naturally formulated questions, and the second one with prompt-tuned questions using the identified prompt patterns. We asked an entrepreneurship educator to evaluate the two sets of answers and gathered feedback from her. The evaluation session was designed as an unstructured interview with open questions. The materials evaluated by the entrepreneurship educator are available at <https://figshare.com/s/0b5588735abbc484098c>.

In the second step, we invited four students at our university who were working on various startup ideas as part of an entrepreneurship course. We set the context of the conversation to *problem validation* (the first phase of startup development based on the Lean Startup methodology [12]) which was the focus of their startup projects at the time of the study. Each of them was asked to interact individually with ChatGPT within the defined context. They first used the questions they formulated naturally and intuitively. Then we helped them repeat the conversation applying the prompt patterns where applicable to their questions. During these sessions, we observed how they formulated questions, interacted with ChatGPT, and reacted to the answers they received. After these sessions, we asked them to evaluate the two sets of answers by commenting freely on them. The documented conversations from these sessions can be found at <https://figshare.com/s/0b5588735abbc484098c>.

All collected ChatGPT conversations, feedback from the entrepreneurship educator and students, as well as our observations will be analyzed systematically

² <https://www.youtube.com/watch?v=bJ8B6hK0pPs>.

³ <https://www.teachingentrepreneurship.org/>.

using appropriate qualitative data analysis techniques. In Sect. 4, we report the preliminary results.

4 Preliminary Results

Table 1 lists the prompt patterns selected from [3] (as described in Sect. 2) and their application in the conversations described in Sect. 3. Some patterns are applied independently, at the beginning of a conversation, or before a group of questions. Others are integrated as part of the questions.

Table 1. Prompt patterns used in startup-related conversations with ChatGPT

Pattern name	When to use	Example of prompt-engineered question (<i>Italic text is the original question</i>)
Persona	When the startup team would like the LLM to assume a specific startup role, e.g., a co-founder, a mentor, or an investor, and provide a detailed answer accordingly.	“Please act as a startup mentor, and answer my following questions.” (This prompt was put at the beginning of a conversation.)
Context Manager	When the startup team would like the LLM to focus on (or exclude) a specific stage or aspect of the startup process.	“Within the scope of startups, please consider only the early phases of startup development when answering my question. <i>What could be the approaches to validate the problems that our startup intends to solve?</i> ”
Flipped Interaction	When the startup team asks about a topic that they have limited knowledge on. This pattern enables the LLM to guide the team to ask more questions to obtain more knowledge on the topic	“Please ask me questions to answer my following question. When you have enough information to answer my question, create an answer to my question with consideration of all information provided to you. <i>Which customer segment should our startup serve?</i> ”
Cognitive Verifier	When the startup team asks a question that is too complex and needs to be decomposed into sub-questions.	“ <i>How could we design our landing page to test the problem-solution fit?</i> Please generate three additional questions that would help you give a more accurate answer to this question. When I have answered the three questions, combine the answers to produce the final answers to my original question.”
Alternative Approaches	When the startup team looks for alternative options and would like to compare them using certain criteria.	“ <i>What is the riskiest assumption for our business model?</i> Please generate alternative answers to this question, and then compare the level of risks involved.”
Question Refinement	When the startup team is unsure what is the right question to ask and would like to get help to rephrase the question.	“From now on, whenever I ask a question, please suggest a better version of the question to use, incorporating information specific to the question that I am using, and check with me if I would like to use the suggested version of the question.” (This prompt was used before a group of questions that need refinement.)
Template	When the startup team would like to ask the LLM to produce structured output applying a given template.	“ <i>Please create a lean canvas based on our startup idea.</i> When doing so, please follow a template that I provide to format your output. The template can be found at https://gustdebacker.com/lean-canvas/ . Please preserve the formatting and overall template.”

It is evident that, based on our initial analysis of the conversations, the prompt-engineered questions tend to elicit more elaborated answers or enhance the conversational interactions and experience. The feedback from the entrepreneurship educator and the four students was generally positive toward the answers to the prompt-engineered questions. They commented that the answers were more specific, had more details, and in some cases less assertive than the answers to their original questions. However, in a few cases, both the educator and students considered the prompt-engineered questions produced worse answers. We need to conduct further analysis to understand whether it is because the prompt pattern applied was not appropriate.

Our initial observation of the interactions between the students and ChatGPT indicates that not all of them were equally comfortable or confident when asking questions to ChatGPT. Some of them were struggling to formulate appropriate questions to ask or ask follow-up questions. This may be partially linked to how familiar they were with conversing with LLMs, and partially attributed to how well they learned *problem validation* as a startup topic. They also revealed different attitudes towards ChatGPT and the answers to some prompt-engineered questions. An interesting comment was made by one student when he received the answer to a question prompt-tuned using the “Flipped Interaction” pattern. He commented that he expected ChatGPT to always provide clear answers rather than asking further questions. In his opinion, answering one question with another or more questions was a sign of not being polite.

5 Discussion

The study was conducted using conversations that are meaningful for early-stage startups. Brainstorming is one of the most intensive activities in the initial phase of a startup. If a startup team intends to use LLMs to support their brainstorming activities, they could apply the prompt patterns that can lead to divergent ideas as well as convergent answers. Several patterns examined in our study have good potential to stimulate divergent thinking. **Flipped interaction** is a good pattern for this purpose. Using this pattern to formulate prompts and tune questions, rather than being provided with direct answers, the startup team can open their minds and ponder on more questions that are relevant to their business idea. This style of interaction is close to the mentoring relationship between a startup team and their mentor. It facilitates the team to think more actively rather than looking for fast and simple answers. Another pattern, **Cognitive Verifier**, can produce a similar effect. It differs from Flipped Interaction in that it does provide a conclusive answer after decomposing a complex question into several sub-questions. The third pattern, **Alternative Approaches**, can also support divergent thinking. Applying this type of prompt to LLMs can return multiple answers/possibilities to a question to which a single answer is not desirable.

The preliminary results indicate that, without applying any prompt engineering techniques, startups may not obtain the desired benefits from LLMs such as

more creative ideas and new insights and understanding of their startup business. Using prompt patterns effectively can help startup teams overcome their lack of knowledge regarding startup processes and deficient prompt engineering skills. The results also serve as a reminder that the successful application of prompt engineering to LLMs is not solely a technical matter. Socio-cultural and human factors will significantly influence the effectiveness of prompt engineering.

There are several limitations in our study. Firstly, we applied the prompt patterns to ChatGPT only. Their applicability to other LLMs remains unclear. Applying and customizing prompt patterns for other LLMs is an interesting future study. Another limitation is that we studied a small number of students who develop startup ideas in a university course setting. Further studies are needed to understand what questions real-world startups are asking LLMs and whether the prompt patterns are equally applicable and useful for them. Lastly, our study was focused on the initial phase of the startup process. Whether the findings are valid for more mature startups is yet to be understood.

6 Conclusion

The recent upsurge of generative AI, including LLMs, opened up numerous exciting opportunities for startups. In this paper, we applied prompt engineering to help startup teams obtain better results when interacting with LLMs. We investigated the application of a set of prompt patterns to ChatGPT. The initial results show that some patterns are more suitable for brainstorming which is a typical activity conducted by early-stage startups. Prompt-tuned questions may lead to more specific and more detailed responses, but it is not guaranteed. In addition, human factors can play an important role in the effective application of prompt patterns.

The study presented in the paper is the first step toward building AI assistants for startups based on LLMs. To reach our goal eventually, we will employ a design science research approach. Two main artifacts are envisioned: 1) a “startup prompt book” in which prompt patterns are a main component. It is a knowledge base that contains a set of rules that transform intuitively expressed questions and requests made by startup teams into prompt-engineered questions as input to LLMs; and 2) a “prompt engine” that can automate the prompt engineering process and choreograph conversations with LLMs. These two artifacts are at the core of an AI assistant that can become a valuable resource for a startup team, as a co-founder, a team member, or a mentor.

To address the limitations mentioned previously, the presented study can be extended to include other LLMs, use a more diverse and representative sample of startups, and explore the applicability of their findings to startups in various stages of development. We believe that the results we obtain in the startup context can be generalisable to established companies for endeavours such as product innovation (e.g., in the ideation phase). However, the generalisability needs to be validated by future research. Last but not least, the potential drawbacks and risks of using LLMs (such as privacy and ethical issues) is an important research topic that needs to be investigated in future research.

References

1. Short, C.E., Short, J.C.: The artificially intelligent entrepreneur: ChatGPT, prompt engineering, and entrepreneurial rhetoric creation. *J. Bus. Ventur. Insights* **19**, e00388 (2023). <https://doi.org/10.1016/j.jbvi.2023.e00388>
2. Liu, P., Yuan, W., Fu, J., Jiang, Z., Hayashi, H., Neubig, G.: Pre-train, prompt, and predict: a systematic survey of prompting methods in natural language processing. *ACM Comput. Surv.* **55**(9), 1–35 (2023)
3. White, J., et al.: A prompt pattern catalog to enhance prompt engineering with ChatGPT (2023). <https://arxiv.org/abs/2302.11382>
4. Giardino, C., Bajwa, S.S., Wang, X., Abrahamsson, P.: Key challenges in early-stage software startups. In: Lassenius, C., Dingsøyr, T., Paasivaara, M. (eds.) *XP 2015. LNBP*, vol. 212, pp. 52–63. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-18612-2_5
5. Nguyen-Duc, A., Hoang, T.N., Bøe, T., Sundbø, I.: Understanding the role of artificial intelligence in digital startups: a conceptual framework (2023). https://www.researchgate.net/profile/Anh-Nguyen-Duc-3/publication/369943717_Understanding_the_Role_of_Artificial_Intelligence_in_Digital_Startups_A_Conceptual_Framework/links/6435aecedad9b6d17dc4ef8b5/
6. Radford, A., Narasimhan, K., Salimans, T., Sutskever, I., et al.: Improving language understanding by generative pre-training (2018). <https://www.mikecaptain.com/resources/pdf/GPT-1.pdf>
7. Dimitrov, M.: What business leaders should know about using LLMS like ChatGPT (2023). <https://www.forbes.com/sites/forbesbusinesscouncil/2023/02/07/what-business-leaders-should-know-about-using-llms-like-chatgpt/>
8. Brown, T., et al.: Language models are few-shot learners. *Adv. Neural. Inf. Process. Syst.* **33**, 1877–1901 (2020)
9. Zamfirescu-Pereira, J., Wong, R., Hartmann, B., Yang, Q.: Why johnny can't prompt: how non-AI experts try (and fail) to design LLM prompts. In: *Proceedings of the 2023 CHI Conference on Human Factors in Computing Systems (CHI 2023)* (2023)
10. Dwivedi, Y.K., et al.: So what if ChatGPT wrote it?" multidisciplinary perspectives on opportunities, challenges and implications of generative conversational AI for research, practice and policy. *Int. J. Inf. Manage.* **71**, 102642 (2023)
11. Zhou, D., et al.: Least-to-most prompting enables complex reasoning in large language models, arXiv preprint [arXiv:2205.10625](https://arxiv.org/abs/2205.10625) (2022)
12. Ries, E.: *The lean startup: How today's entrepreneurs use continuous innovation to create radically successful businesses*. Currency (2011)





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.





ChatGPT as a Fullstack Web Developer - Early Results

Pekka Abrahamsson¹ , Tatu Anttila¹, Jyri Hakala¹, Juulia Ketola¹, Anna Knappe¹ , Daniel Lahtinen¹, Väinö Liukko¹, Timo Poranen¹  , Topi-Matti Ritala¹, and Manu Setälä²

¹ Tampere University, Tampere, Finland

² Solita Ltd., Helsinki, Finland

Abstract. The arrival of ChatGPT has caused a lot of turbulence also in the field of software engineering in the past few months. Little is empirically known about the capabilities of ChatGPT to actually implement a complete system rather than a few code snippets. This paper reports the first-hand experiences from a graduate level student project where a real-life software platform for financial sector was implemented from the scratch by using ChatGPT for all possible software engineering tasks. The main conclusions drawn are as follows: 1) these findings demonstrate the potential for ChatGPT to be integrated into the software engineering workflow, 2) it can be used for creating a base for new components and for dividing coding tasks into smaller pieces, and 3) noticeable enhancements in ChatGPT-4, compared to ChatGPT-3.5, indicate superior working memory and the ability to continue incomplete responses, thereby leading to more coherent and less repetitive dialogues.

Keywords: AI assisted · software development · software engineering · AI programming · ChatGPT · large language models · artificial intelligence

1 Introduction

The introduction of ChatGPT into the landscape of technology has generated a notable amount of disruption, especially within the field of software engineering. However, despite this growing interest, empirical knowledge about the actual capabilities of ChatGPT remains limited. This lack of comprehensive understanding is particularly evident when considering the potential of ChatGPT to design and implement holistic systems as opposed to merely generating discrete fragments of code. There exists a significant difference between crafting isolated code snippets and deploying a fully realized software solution, a distinction that is yet to be thoroughly explored in the context of ChatGPT.

This article describes a student software project that was created to explore the use of artificial intelligence (AI) in software development. The main goal of the project was to investigate the effectiveness of ChatGPT in practice.

© The Author(s) 2024

P. Kruchten and P. Gregory (Eds.): XP 2022/2023 Workshops, LNBP 489, pp. 201–209, 2024.

https://doi.org/10.1007/978-3-031-48550-3_20

Overall, this project contributes to the field of AI-assisted software development by providing valuable experience of using ChatGPT as tool in software development. The remainder of this article provides related research in Sect. 2, a detailed description of the project and the research design in Sect. 3, results in Sect. 4, and conclusions.

2 AI Assisted Software Development

Artificial Intelligence (AI) is a branch of computer science that focuses on creating intelligent machines that can perform tasks that typically require human intelligence, such as understanding natural language, recognizing patterns, making decisions, and solving problems.

One type of AI tool that has gained significant attention in recent years is the large language model (LLM) like OpenAI's GPT-4 [7]. LLM is AI model that is trained on massive amounts of data to generate human-like text output. GPT-4 uses transformer-style model to predict the content and structure of text based on an input, usually text as well. This can be used in a wide range of natural language processing tasks, such as language translation, text summarization, and answering questions.

ChatGPT has provided a chatbot interface for interacting with OpenAI's GPT-models [2]. This has made the capabilities of LLMs more widely known, which in turn has sparked the research around use cases for this technology. Current research include studies related to prompt patters [13], human-bot collaborative architecting [1] and using ChatGPT for programming numerical methods [6]. Treude [11] has developed a prototype to compare different GPT model solutions, and Dong and others [4] developed a self-collaboration code generation framework. Surameery and Shakor [10] have applied ChatGPT to solve programming bugs.

3 Research Design

In this section we introduce the project background and project implementation details including the implemented features.

3.1 Project Background

Solita Ltd. [8] is a large software consultancy company in the Nordic countries. Solita collaborates with universities by inventing exercise topics and supervising student exercises. They challenged the student team to undertake an AI-assisted, large-scale project.

Project topic was chosen from well-defined public procurement requests on the Hilma portal [5], a website for procurement in the Finnish public sector. It was agreed in the project that the specifications would not be directly used as input material for AI, but the prompts given to AI were written mostly by the

team themselves. However, the number of fields in the user interface was kept the same as in the original request, etc.

The selected project, Valvontatyöpöytä (VTP) is a platform for financial supervision, designed to support the operations of an organization. The intended user group for the VTP is financial professionals, including supervisors, managers, and analysts.

3.2 Project Implementation

The VTP project [12] was proposed in the end of December 2022. The project was then accepted by a seven member team. The project started in the end of January 2023. The team consists of three master’s level students and four Bachelor’s level students of Computer science or Information technology. None of the team members had earlier experience on AI assisted software development. General overview of the course’s practices and schedule are described by Sten and others [9]. First was sprint 0 (one week) to plan the project and set up the development environment, then five two-week implementation sprints, and then one week quality assurance sprint. The total duration of the project is 15 weeks. Sprint 5 and the QA sprint are not covered in this report as the project is ongoing. Project phases and implemented features per sprint are show in Fig. 1. The team utilized a Kanban board to manage tasks and issues.

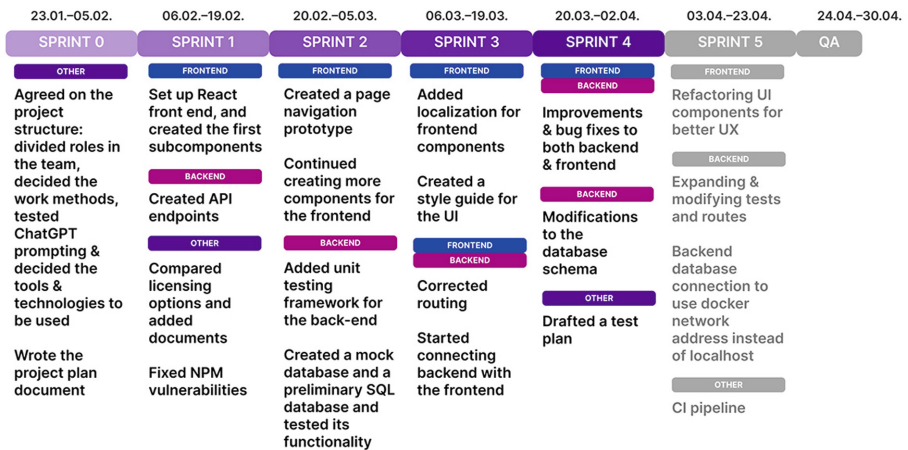


Fig. 1. Project phases and implemented features per sprint.

3.3 Development Environment

The development environment was built piecemeal by consulting ChatGPT on suitable technologies for the projects needs. The team fed ChatGPT prompts

explaining what they were currently trying to accomplish and ChatGPT replied with multiple recommendations. The team then cherry-picked from the recommendations based on their own preferences and previous experience. The effect of picking technologies with such a process is twofold. Firstly, ChatGPT is more likely to recommend technologies it has knowledge of. Secondly, it makes the teams' efforts in reviewing the generated code easier.

Table 1. Technical implementation environment and technology selection criteria.

Item	Description	Selection criteria
Language	JavaScript	Recommended by ChatGPT
Language	HTML and CSS	Recommended by ChatGPT
Language	Shell	Developer decided
Containerization	Docker	Customer requirement
Database	MySQL	Recommended by ChatGPT
AI assistant	ChatGPT 3.5 and 4.0	Customer requirement
Front-end framework	React	Recommended by ChatGPT
Back-end framework	Node.JS, Express	Recommended by ChatGPT
Test framework	Mocha, Chai	Recommended by ChatGPT
Version control system	GitHub	Team decided
Licence	MIT	Recommended by ChatGPT

Table 1 lists technologies used in the project with the corresponding selection criteria. As seen in Table 1 there are exceptions on which ChatGPT was not consulted at all. These include using ChatGPT as the sole AI assistant, and containerization of the produced application, both of which were requirements laid out by the customer. The team decided to use GitHub as their version control system to enable easy collaboration. The only exception in languages used in the project comes from a Bash script used by the backend container to wait for the database to fully initialize before trying to establish a connection.

3.4 Development Process

The team's process of working with ChatGPT is illustrated in Fig. 2. When a task was related to existing code, the assigned team member would provide the relevant code to ChatGPT and request it to generate a solution. If the task was not related to existing code, the team member would first ask ChatGPT for recommendations before requesting it to produce the code. Once ChatGPT generated the code, the team member would review it for correctness. If the code was deemed satisfactory, the team member would add it to the code base, save the chat session as a markdown file, and create a pull request with the chat as an attachment. If the code was not acceptable, the team member would provide

the problematic code back to ChatGPT and repeat the process, iterating until a satisfactory solution was achieved or asking for a new recommendation for another approach.

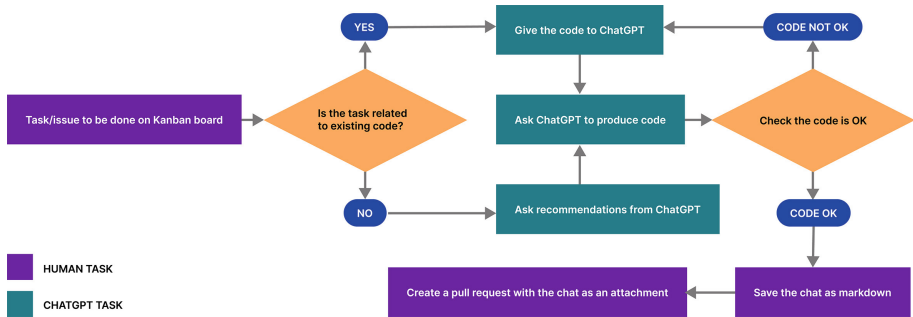


Fig. 2. Process of working with ChatGPT.

The project team logged their weekly work hours according to different categories (Documentation, Requirements, Design, Implementation, Testing, Meetings, Studying, Other, Lectures). After sprint 4 the project had a total 607 h logged. Throughout the project, the project team met with the customer twice a week to ensure quality assurance and planning were on track. This is a reason why meetings (198 h, 33%) have such a considerable part in the logged hours. So far the implementation has taken 131 h (22%) and studying 116 h (19%). The logged time also included university course related subjects, such as lectures and other studying, which did not relate directly to the implementation of the project.

3.5 Implemented Features

Based on ChatGPT’s suggestion, we began by setting up the foundation for our React JavaScript project. Implemented features per sprints are shown in Fig. 1. Following the wireframes in the original procurement requests, we commenced implementing the Inspection Information page shown in Fig. 3. We adopted a component-by-component approach to building the application, and once the first frontend components were completed, we proceeded to develop the backend and database infrastructure.

To better comprehend the application’s functionality, we utilized the wireframe images to create a prototype. In our development process, we proceeded to implement a new view for the application, the Inspection Plan page. Additionally, we created localization possibility into the application, allowing for text to be displayed in both Finnish and English. With the assistance of ChatGPT, we created a style guide for the UI and began implementing it into the application.

We proceeded to build the correct routing and to integrate the frontend and backend together. Recognizing the need for adjustments to the database

Voitot	Alkuperäinen pää	Tehdys pää	Kommentit	Liikkeen dokumentit
Tarkastusajankohdat	01/01/2022		This is the first target timeframe	Liikkeen dokumentit
Goal B	03/01/2022	03/02/2022	This is the second target timeframe	
Goal C	03/01/2022	03/02/2022	This is the third target timeframe	
Goal D	04/01/2022	04/02/2022	This is the fourth target timeframe	
Goal E	05/01/2022	05/02/2022	This is the fifth target timeframe	
Goal F	06/01/2022	06/02/2022	This is the sixth target timeframe	
Goal G	07/01/2022	07/02/2022	This is the seventh target timeframe	
Goal H	08/01/2022	08/02/2022	This is the eighth target timeframe	
Goal I	09/01/2022	09/02/2022	This is the ninth target timeframe	
Goal J	10/01/2022	10/02/2022	This is the tenth target timeframe	
Goal K	11/01/2022	11/02/2022	This is the eleventh target timeframe with target_id 2	
Goal L	12/01/2022	12/02/2022	This is the twelfth target timeframe with target_id 2	

Tarkastuksen tiedot	Viimeisimmät asiakirjat																				
<table border="1"> <thead> <tr> <th>Tarkastuksen tiedot</th> <th>Tarkastuksen tiedot</th> </tr> </thead> <tbody> <tr> <td>Alku</td> <td>2022-01-01 00:00:00</td> </tr> <tr> <td>Viimeisin</td> <td>2022-01-01 00:00:00</td> </tr> <tr> <td>Kokonaissumma</td> <td>2022-01-01 00:00:00</td> </tr> </tbody> </table>	Tarkastuksen tiedot	Tarkastuksen tiedot	Alku	2022-01-01 00:00:00	Viimeisin	2022-01-01 00:00:00	Kokonaissumma	2022-01-01 00:00:00	<table border="1"> <thead> <tr> <th>Asiakirjat</th> <th>Käsitteet</th> <th>Merkki</th> </tr> </thead> <tbody> <tr> <td>Document 1</td> <td>John Doe</td> <td>20/01/2022 02:00</td> </tr> <tr> <td>Document 2</td> <td>John Doe</td> <td>20/01/2022 02:00</td> </tr> <tr> <td>Document 3</td> <td>John Doe</td> <td>20/01/2022 02:00</td> </tr> </tbody> </table>	Asiakirjat	Käsitteet	Merkki	Document 1	John Doe	20/01/2022 02:00	Document 2	John Doe	20/01/2022 02:00	Document 3	John Doe	20/01/2022 02:00
Tarkastuksen tiedot	Tarkastuksen tiedot																				
Alku	2022-01-01 00:00:00																				
Viimeisin	2022-01-01 00:00:00																				
Kokonaissumma	2022-01-01 00:00:00																				
Asiakirjat	Käsitteet	Merkki																			
Document 1	John Doe	20/01/2022 02:00																			
Document 2	John Doe	20/01/2022 02:00																			
Document 3	John Doe	20/01/2022 02:00																			

Fig. 3. Inspection information in VTP.

schema, we initiated revisions while concurrently addressing application styling and adding several popup forms.

4 Results

In this section we observe ChatGPT discussions, code and lessons learned.

4.1 Discussions with ChatGPT

We stored ChatGPT conversations that were relevant to the project in Markdown format. The resulting Markdown files were included into the description part of pull requests made to the project GitHub repository.

The conversation length varies a lot depending on how big changes were requested and if any problems occurred. From sprint 0 to sprint 4, 39 pull requests were merged to the codebase. In these conversations the number of prompts made ranged from 1 to 129. On average a pull request had 19 prompts. A common way to start conversation with ChatGPT was to describe the problem or wanted feature or to paste existing code and ask for needed changes. If the AI model couldn't give a solution directly based on the first prompt, the developer would give more information or clarifications in an iterative manner where the solution was found by fine-tuning the earlier answers.

4.2 Code

The application implements a basic three-tier architecture, where the front-end communicates with the back-end through a REST API. Each tier runs in their own Docker container and the whole system is managed with Docker Compose.

Table 2 provides a breakdown by type of file and line for the project's code base. File and blank, comment and code line counts were calculated from the

project’s repository using cloc [3]. The vast majority of lines shown are a result of bootstrapping a React project to be used as the front-end client. This includes most of the JSON lines that node package manager uses for managing dependencies.

However, the team managed to use ChatGPT to generate all of the code directly related to running the project. This includes containerization (both Dockerfiles, a YAML file), CI pipeline (other YAML file), database initialization clauses (SQL), and the entire RESTful API that makes up the project’s back-end, including unit tests for its routes. Furthermore, all React components, their layout and styling (CSS) were created by ChatGPT based on the teams instructions. The HTML files contain a style guide. In total, 4000 lines of code were generated by ChatGPT.

Table 2. Output of the cloc [3] package based on the project’s repository as of 2.4.2023.

Language	files	blank	comment	code
JSON	6	0	0	18932
JavaScript	36	259	68	2798
CSS	15	149	22	796
HTML	3	42	30	357
Bourne Shell	1	12	6	164
Markdown	4	51	0	82
SQL	1	12	0	73
YAML	2	9	0	58
Dockerfile	2	14	3	20
SVG	1	0	0	1
SUM:	71	548	129	23281

In general, ChatGPT produces code that appears to be relatively high quality. The code mostly works, it is laid out in a logical manner, and has well named variables that make it easy to understand. Its two biggest pitfalls are consistency and attention to detail. The former shows as stylistic differences in blocks of code produced in separate replies by ChatGPT, which were sometimes incompatible to the point of not functioning. The latter problem was mainly encountered when attempting to fit pieces of the project together as it grew more complex which meant conversations with ChatGPT needed more context to be provided. The team noticed marked improvement in both areas when using GPT-4 over GPT-3.5.

4.3 Lessons Learned

ChatGPT has both strengths and limitations in assisting with software development tasks. While it can be helpful in creating a base for new components,

fine-tuning and getting the final result may take longer. It may struggle with updating code consistently, remembering to make changes, and handling errors. Additionally, it can misunderstand questions, have issues with non-determinism, and suffer from token limitations in answers. However, breaking requests into smaller pieces, providing clear descriptions, and using various techniques to avoid length restrictions can improve its effectiveness.

One team member described experience of working with ChatGPT like a rubber ducking, but the duck actually responds. In similar vein to rubber ducking, the output from the language model depends on how well the developer is able to express the problem at hand. A Clear well defined prompt would return better code than a prompt by someone with only a vague idea of what they were trying to achieve. Good development practises still apply, even if the code is written by AI.

5 Conclusions

This paper presents first-hand experiences of using ChatGPT to develop a full-stack software application. Overall, this study contributes to the growing body of literature on the application of language models in software engineering. This study also provides insights for researchers and practitioners interested in exploring the use of ChatGPT for developing real-world software systems.

Based on early results from this exploratory study, the main conclusions drawn were as follows: 1) these findings demonstrate the potential for ChatGPT to be integrated into the software engineering workflow, 2) it can be used for creating a base for new components and for dividing coding tasks into smaller pieces, and 3) noticeable enhancements in ChatGPT-4, compared to ChatGPT-3.5, indicate superior working memory and the ability to continue incomplete responses, thereby leading to more coherent and less repetitive dialogues.

Next steps in the research will include reporting experiences from the remaining sprints, test the application systematically, analyse code quality, and compare ChatGPT generated code with the code written by the developers.

References

1. Ahmad, A., Waseem, M., Liang, P., Fehmideh, M., Aktar, M.S., Mikkonen, T.: Towards human-bot collaborative software architecting with ChatGPT. arXiv preprint [arXiv:2302.14600](https://arxiv.org/abs/2302.14600) (2023)
2. ChatGPT. <https://chat.openai.com/> (2023). Accessed 6 Apr 2023
3. cloc - Count lines of Code. <https://github.com/AIDanial/cloc> (2023). Accessed 13 Apr 4 2023
4. Dong, Y., Jiang, X., Jin, Z., Li, G.: Self-collaboration code generation via Chatgpt. arXiv preprint [arXiv:2304.07590](https://arxiv.org/abs/2304.07590) (2023)
5. Hilma - Public procurement. <https://www.hankintailmoitukset.fi/en/> (2023). Accessed: 31 Mar 2023
6. Kashefi, A., Mukerji, T.: ChatGPT for programming numerical methods. arXiv preprint [arXiv:2303.12093](https://arxiv.org/abs/2303.12093) (2023)

7. OpenAI: GPT-4 Technical Report. arXiv preprint [arXiv:2303.08774](https://arxiv.org/abs/2303.08774) (2023)
8. Solita company. <https://www.solita.fi/en/company/> (2023). Accessed 31 Mar 2023
9. Sten, H., Ahtee, T., Poranen, T.: Evaluation of students' capstone software development projects. In: SEFI Annual Conference, pp. 531–540 (2018)
10. Surameery, N.M.S., Shakor, M.Y.: Use Chat GPT to solve programming bugs. Int. J. Inf. Technol. Comput. Eng. (IJITC) **3**(01), 17–22 (2023). ISSN: 2455-5290
11. Treude, C.: Navigating complexity in software engineering: a prototype for comparing GPT-n solutions. arXiv preprint [arXiv:2301.12169](https://arxiv.org/abs/2301.12169) (2023)
12. VTP - Source code repository for the Valvontatyöpyytä. <https://github.com/AI-Makes-IT/VTP> (2023). Accessed 31 Mar 2023
13. White, J., Hays, S., Fu, Q., Spencer-Smith, J., Schmidt, D.C.: ChatGPT prompt patterns for improving code quality, refactoring, requirements elicitation, and software design. arXiv preprint [arXiv:2303.07839](https://arxiv.org/abs/2303.07839) (2023)

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-Quantum SE 2023



Reviewing Crypto-Agility and Quantum Resistance in the Light of Agile Practices

Lodovica Marchesi^(✉) , Michele Marchesi , and Roberto Tonelli 

Department of Mathematics and Computer Science, University of Cagliari, Cagliari,
Italy

{lodovica.marchesi,marchesi,roberto.tonelli}@unica.it

Abstract. The term crypto-agility means the ability to quickly and securely change cryptographic algorithms and related data, in the case of their compromise. In this context, the advent of quantum computing constitutes a new paradigm, which poses existential threats to current cryptographic algorithms. Even if these attacks are not an imminent danger, we must be prepared to change the cryptographic algorithms at risk with new, quantum resistant ones. This is by no means an easy task, because cryptographic algorithms are used everywhere and are often also implemented on the hardware. In this paper, we analyze the similarities and the differences between traditional agility and crypto-agility, and investigate the prospects of using agile and lean practices in the context of crypto-agility to introduce quantum resistant algorithms. In particular, for the main agile and lean practices we discuss if and how they can be useful for obtaining crypto-agility. We also investigate how the features key to crypto-agility can be helped by the agile and lean approach.

Keywords: Agile methods · Cryptographic agility · Encryption algorithms · Quantum resistance

1 Introduction

The introduction of agile methodologies in the late 1990s and the term “agile” itself in the 2001 Agile Manifesto had a deep impact on software engineering and computer science [1]. In a short time, “agile” became a buzzword used wherever you had to emphasize the ability to respond quickly and well to challenges and requirement changes.

A few years after the Agile Manifesto, people in the cybersecurity community introduced the term “cryptographic agility” or “crypto-agility”. This term was defined first in a paper by LaMacchia and Manferdelli in 2006 [9]. Their paper presented the Microsoft’s new core cryptographic API, “Crypto Next Generation” (CNG), which was claimed to have “cryptographic agility”. In the context of CNG, crypto-agility means the ability to change its cryptographic algorithms (CAs) and related data (setting parameters, key storage, etc.) in the case of their compromise, in a quick a secure way.

© The Author(s) 2024

P. Kruchten and P. Gregory (Eds.): XP 2022/2023 Workshops, LNBP 489, pp. 213–221, 2024.

https://doi.org/10.1007/978-3-031-48550-3_21

In fact, practical cases in which CAs had to be changed quickly to avoid damage due to their compromise are very rare. However, the advent of quantum computing constitutes a new paradigm which poses new challenges to cryptography because robust quantum computers (QCs) have been proven to be able to break several important CAs currently used. Important milestones still remain before a marketable and truly usable QC can exist to solve real-world problems. The most optimistic experts estimate that it will take 5 to 10 years from now to build a viable QC. The more cautious ones predict 15 to 30 years [13].

We know that agility means the ability to react quickly to changes. Agile and lean software development were introduced to support traditional programming in projects where timing is essential. Crypto-agility for quantum resistance (QR), on the other hand, typically deals with changes that likely will be mandatory at the end of this decade, thus operating in time intervals completely different.

In this paper, we analyze the links, the similarities, and the differences between crypto-agility and traditional agility. We discuss which agile and lean principles and practices are still relevant to cryptographers who are in charge of finding and substituting traditional algorithms with new quantum-resistant ones and which are not relevant to them.

The rest of the paper is organized as follows: Sect. 2 provides a short review of the literature; Sect. 3 presents the concept of crypto-agility, highlighting its main properties, while Sect. 4 explains what QC is and what QR means. In Sect. 5 we describe the relationship between crypto-agility for QR with agile and lean practices, to dive into how agile software development can help the development, testing, and deployment of new quantum resistant algorithms.

2 Related Work

The first use of the term “crypto-agility” was made in 2006 in [9]. After that date, the term compares in technical reports of standard working groups (IETF, ETSI, NIST). Only from 2018 onwards, also due to the efforts to choose and standardize quantum resistant algorithms, which were announced at PQCrypto conference in 2016, the term has been used and discussed in many forums. The related wikipedia page also dates back to November 2018.

In 2019 Grote et al. described the strategy of post-quantum cryptography and crypto-agility. They reviewed the proposed quantum resistant algorithms and highlighted the need for crypto-agility to effectively replace non-quantum resistant CAs with quantum resistant ones [5]. In 2021 Mashatand and Heintzman recommended a crypto-agile process to assess and mitigate the exposure of organizations to quantum attacks. The proposed process includes steps such as *Determine Transition Path, Wait for Standardization, Invest in Crypto-agility, Establish and Maintain a Quantum-Resistance Roadmap, Implement Hybrid Cryptography* [11]. In the same year, Ma et al. proposed CARAF: Crypto Agility Risk Assessment Framework [10]. CARAF is aimed at analyzing and evaluating the risk that results from the lack of crypto agility, in order to determine an appropriate mitigation strategy commensurate with the risk tolerance. The application

of this framework was demonstrated with a case study regarding QR. Furthermore, Zhang and Miranskyy analyzed the threats posed by QCs and compared the related mitigation strategies to those used to address the Y2K bug [18]. They proposed a road map for software developers to address encryption-related challenges associated with quantum attacks, especially using crypto-agility. In 2022, Holm et al. proposed the Crypto-Agility Maturity Model (CAMM) to determine the state of crypto-agility of a given software or IT landscape and to improve it [7]. CAMM consists of five levels named: (0) Initial/Not possible, (1) Possible, (2) Prepared, (3) Practiced, and (4) Sophisticated. For each level, a set of requirements is formulated based on literature review. Initial feedback from field experts confirmed that CAMM has a well-designed structure and is easy to understand.

2.1 Agility and Cybersecurity

There are several studies on how agile practices can be made compatible with security assurance, starting from the seminal work of Bezsonov [2] on merging XP and security practices. Among these studies, two main areas emerge: updates to Scrum and in general to agile processes to manage security aspects, and user stories to specify security requirements.

Regarding the first area, Fitzgerald et al. identified the main incompatibility issues between agile characteristics and constraints imposed by regulated environments and illustrated through a detailed case study how an agile approach can be implemented successfully in a regulated environment [3]. Ghani et al. suggested adding Security Backlog and the role of a Security Master in Scrum [4]. Othmane et al. proposed a method to ensure the security of software increments, integrating security engineering activities into the agile software development process [14].

Regarding security requirements, Kongsli proposes the concept of “misuse story”, derived from the “misuse case”, and reports on experiences with the use of misuse stories and automatic security tests in the development of web applications [8]. Williams et al. suggested the Protection Poker game to find the relative security risk of each requirement [17]. For a recent and comprehensive survey of security in agile software development, we refer the reader to the work of Rindell et al. [15].

In all cited papers, the relationships between the agile approach and general cybersecurity are considered, including protection against multiple forms of cyber attacks. In this paper, we are interested to practices related to the development and integration of cryptographic methods and tools into software systems, which is only one of the several aspects of cybersecurity. We deal with security requirements of cryptographic tools related to asymmetric cryptography, document encryption and decryption, digital signatures, key storage and the like.

3 Definition of Crypto-Agility

Crypto-agility is strictly related to cybersecurity, but its scope is much narrower. As reported above, it regards the ability to easily change the algorithm implementations used by cryptographic protocols and to provide a high level of abstraction by the API for core cryptographic operations [9].

The key properties of crypto-agility were provided by Mehrez and El Omri [12]. They state that “*crypto-agility is the ability of a system to migrate easily from one CA to another, in a way that is flexible, scalable, and dynamic*”. The most important crypto-agility properties among those reported by them are:

- **Extensibility:** ability to add new algorithms or new parameters to the system as efficiently as possible.
- **Removability:** ability to gracefully retire cryptographic systems that have become vulnerable or obsolete.
- **Fungibility:** ease to swap security components; also, the ability for machines to select their security algorithms in real time and based on their combined security functions;
- **Interoperability:** Crypto-Agility solutions must be interoperable between independent implementations based purely on the information provided in the specification.
- **Updateability:** support of secure updates or patches of CAs in the system.
- **Compatibility:** if we replace software on a system, the new software modules and patches should be able to operate on the same hardware.
- **Reversibility:** if any software update fails, the system should be able to return to the previous working software version.

The object of crypto-agility is the development or updating of software systems that use CAs. These systems are typically used for secure data transmission and for the storage and retrieval of encrypted data. Another field impacted by the change in CAs is that based on a blockchain, which makes extensive use of asymmetric cryptography.

The CAs which are actually used follow standards enacted by various organizations, among which the most prominent is NIST. There are standard CAs for symmetric and asymmetric encryption, hash functions, key management. The standards include detailed specification of CA libraries API, so that systems using different libraries can exchange encrypted data, provided that these libraries follow the API specifications.

Typically, large, regulated organizations are more impacted by changes in CAs than small ones. Since CAs are largely regulated, including their APIs, the software that actually needs to be written is the software that uses these CAs to encrypt (maybe using more than one CA), send or store data, and decipher them.

4 Quantum Computing and Quantum Resistance

Quantum computing represents a significant breakthrough in computer science and will have a strong impact on many fields, such as science, finance, artificial intelligence, pharmacology, and many others. Unfortunately, it also has the power to breach current cryptography systems, among which secure Internet communications, digital signatures, digital currencies, and digital ledger technology (DLT). The cryptography used in these systems is composed of *Hash functions*, which guarantee immutability of data, and in blockchains are also applied in proof of work; *Symmetric cryptography*, used to encode and decode information that must remain confidential; *Asymmetric cryptography*, which is behind SSL/TLS protocol ensuring secure Internet communication, and in DLT guarantees the propriety of the assets linked to an address.

No classic computer in existence is capable of performing calculations fast enough to reverse this math in any usable time frame. However, the advent of QC constitutes a new paradigm which poses new challenges to cryptography. Important milestones still remain before a marketable and truly usable QC can exist to solve real-world problems. Experts estimate that it will take 5 to 30 years from now to build a viable QC [13].

When QCs will become operational, the currently understood menace they will pose to cryptography is based on Shor's algorithm for quickly factoring the product of two very large primes [16]. This algorithm will allow one to unhinge the RSA algorithm, and with a small variant also the ECDSA algorithm. These algorithms are currently the most used for asymmetric cryptography. Today, a digital computer that uses the most efficient algorithm known would carry out the factoring of a number of 300 digits in about 150,000 years. A QC using the Shor algorithm would find the solution in seconds.

Another threat is Grover's algorithm, which allows you to speed up the search for possible solutions to unhinge symmetric encryption algorithms (AES, DES, hash algorithms) and can reduce the difficulty of the problem from n to \sqrt{n} [6]. However, the performance of Grover's algorithms is not as innovative and dangerous as for Shor's one because it can be easily countered by increasing the length of the encryption key.

Luckily, several CAs that are quantum resistant already existed in the nineties, and more have been introduced after the discoveries of Shor and Grover. Standardization bodies such as NIST (National Institute of Standards and Technology) and ETSI (European Telecommunications Standards Institute) have been working on standard quantum resistant algorithms for almost a decade, and a set of international standards is expected in 2025 [11].

One of the main issues faced by organizations to be prepared for quantum menace, is the fact that CAs are ubiquitous in the software and hardware systems used. Therefore, migrating to quantum resistant solutions will not be an easy journey. To this end, an organization can take advantage of a process that exhibits the properties of crypto-agility as defined in Sect. 3.

5 Can Agility Help Crypto-Agility?

Agile and Lean software development was introduced to shorten development times and accommodate changes, without compromising on quality. Its time-frames, depending on the specific activities, vary from hours to days or weeks.

Mimicking the well-known approach to software development, crypto-agility means the ability to react to CA changes effectively and timely. However, the need to protect a system from quantum attacks likely will be mandatory not before the end of this decade, thus its time-frame is of the order of years, or even decades.

We asked ourselves if agile principles and practices can be successfully used in the context of crypto-agility, and specifically to effectively address the development and adoption of quantum resistant software. To answer this question, we consulted some software practitioners working in the field of cybersecurity, and also took advantage of our experience in studying QR for blockchain applications. The research questions asked were: (i) “How suitable are the agile and lean practices to support crypto-agility?”; and conversely (ii) “How crypto-agility features can benefit from the agile approach?”.

The context is the development of software that must strictly follow standards regarding its CA and API, and that uses standard libraries to send, store, and receive encrypted data, to decode them, to manage keys, and to combine CAs to obtain stronger security or to manage digital signatures and secure ownership of digital assets. Following a crypto-agile principle, this software should also be able to automatically choose the “right” CA, and manage the updating of CA libraries.

Table 1 reports the main Agile and Lean practices, with a judgment on their relevance to crypto-agility. You can see that most of these practices were deemed to be useful, or very useful. The requirements should be expressed as features, because most of them do not regard direct interaction with an user. Automated testing is of the utmost importance. Tests can also be defined before coding the CA and other software. We stress that the number of tests needed to assess the security of the developed software is much higher than in other systems.

Regarding Lean practices, most of them are very useful to apply also to this type of development. The “optimize the whole” practice may not be relevant to this kind of development because the standardized CAs are already defined. However, the choice of which specific quantum resistant CA to use should be carefully made because these CA have memory and CPU requirements much higher than traditional ones. The WIP limitation can, of course, be applied, but here the need of a continuous flow of delivered working features is uncommon, so this practice is not very relevant in most cases.

Regarding the second question “How crypto-agility can benefit from agility?”, Table 2 summarizes the results of our study. For the sake of brevity, we are not able to discuss these results, but they are quite self-explanatory.

Table 1. Suitability of Agile and Lean practices for Crypto-Agility to obtain QR.

Agile practice	Suitability to Crypto-Agility	Agile practice	Suitability to Crypto-Agility
Iterations	Useful, on a longer timescale	Simple design	Useful, though not always applicable to algorithms
Customer-oriented approach	Very useful	Refactoring	Very useful
Product backlog	Very useful	Coding standards	Very useful
User stories or features (MMF)	See discussion	Collective code ownership	Each artifact should have an accountable owner
Agile roles	Useful, a Security Master can be added	Lean practice	Suitability to Crypto-Agility
Timeboxing	Useful, on a longer timescale	Eliminate waste	Useful, on a longer timescale
Scrum daily meetings	Useful	Build quality in	Very useful
Sprint meetings	Useful to plan and assess Sprint work	Create knowledge	Very useful
Retrospective meetings	Very useful to steer the project and upgrade the process	Defer commitment	Very useful
Daily integration	Frequent integration on a longer time scale (not daily)	Deliver fast	Useful, on a longer timescale
Test-driven development (TDD)	See discussion	Respect people	Very useful
Automated tests	Very useful	Optimize the whole	See discussion
Burndown chart	Useful, on a longer timescale	Visualize the Workflow	Very useful
Requirement prioritization	Very useful	Limit Work in Progress (WIP)	See discussion
Pair programming	Maybe for critical tasks	Cumulative flow diagram	Not very relevant

Table 2. Crypto-Agility principles and agile practices

Crypto-Agility Principles and Features	Relationship with Agility and notes	Source
Extensibility: ability to easily change the algorithm implementations used by cryptographic protocols. Non quantum resistant cryptographic algorithms will be replaced QR proved algorithms.	Agility is aimed to promote change, so most agile practices are key to obtain this principle.	[5], [9], [12]
Protocols with a high level of abstraction of the API and modular implementations to easily accommodate the insertion/updating/interoperability of new algorithms.	This feature allows to follow Updateability and Interoperability principles. It requires good design capabilities, so it is linked with the agile and lean practices of simplicity, continuous testing, build quality in, and create knowledge.	[9], [12]
Ability to adjust to select the security algorithms in real time without any or with as little as possible human intervention.	This is Fungibility feature. It is a requirement linked to good design, as above.	[7], [12]
Removability, ability to easily retire cryptographic systems that have become either vulnerable or obsolete.	It is a requirement linked to good design, as above.	[12]
Testing and validation should support all steps of cryptographic processes from implementation to roll out	Clearly linked to the agile practice of continuous, automated testing, and of refactoring. The number of needed tests is much higher than usual.	[7]
Compatibility, if cryptographic software is replaced with new software modules, it should be able to operate smoothly on the same hardware.	Linked to good design, modularity and continuous testing.	[12]
Reversibility, if any software update is not successful, the system should be able to return to the previous working software version.	This feature requires the use of a configuration management system, which is key also for continuous integration and testing.	[12]

6 Conclusions

The advent of QC bring about new risks for cryptographic algorithms, which we may have to deal with within a decade. Being prepared to quickly and safely transform into quantum resistant algorithms is not a simple task, also due to the wide diffusion of cryptographic algorithms both at the hardware and software level. In this article, we have analyzed the similarities and differences between traditional agility and crypto-agility. We investigated whether and to what extent leading agile and lean practices are suited to support crypto-agility. Most have been rated useful or very useful, with an emphasis on automated and massive testing.

This work is part of a broader involvement of our research group in the field of quantum software engineering.

References

1. Beck, K., Beedle, M., Van Bennekum, A., et al.: The agile manifesto (2001)
2. Beznosov, K.: Extreme security engineering: on employing XP practices to achieve 'good enough security' without defining it (2003)
3. Fitzgerald, B., Stol, K.J., O'Sullivan, R., O'Brien, D.: Scaling agile methods to regulated environments: an industry case study (2013)
4. Ghani, I., Azham, Z., Jeong, S.R.: Integrating software security into agile-scrum method. *Trans. Internet Inf. Syst.* **8**(2), 646–663 (2014)
5. Grote, O., Ahrens, A., Benavente-Peces, C.: Paradigm of post-quantum cryptography and crypto-agility: strategy approach of quantum-safe techniques (2019)
6. Grover, L.K.: Quantum mechanics helps in searching for a needle in a haystack. *Phys. Rev. Lett.* **79**, 325–328 (1997)
7. Hohm, J., Heinemann, A., Wiesmaier, A.: Towards a maturity model for crypto-agility assessment. arXiv preprint [arXiv:2202.07645](https://arxiv.org/abs/2202.07645) (2022)
8. Kongsli, V.: Towards agile security in web applications (2006)
9. LaMacchia, B.A., Manferdelli, J.L.: New vistas in elliptic curve cryptography. *Inf. Secur. Tech. Rep.* **11**(4), 186–192 (2006)
10. Ma, C., Colon, L., Dera, J., Rashidi, B., Garg, V.: CARAF: crypto agility risk assessment framework. *J. Cybersecur.* **7**(1) (2021)
11. Mashatan, A., Heintzman, D.: The complex path to quantum resistance: is your organization prepared? *Queue* **19**(2), 65–92 (2021)
12. Mehrez, H.A., El Omri, O.: The crypto-agility properties (2018)
13. Mosca, M.: Cybersecurity in an era with quantum computers: will we be ready? *IEEE Secur. Priv.* **16**(5), 38–41 (2018)
14. Othmane, L.B., Angin, P., Weffers, H., Bhargava, B.: Extending the agile development process to develop acceptably secure software. *IEEE Trans. Dependable Secure Comput.* **11**(6), 497–509 (2014)
15. Rindell, K., Ruohonen, J., Holvitie, J., Hyrynsalmi, S., Leppänen, V.: Security in agile software development: a practitioner survey. *Inf. Softw. Technol.* **131**, 106488 (2021)
16. Shor, P.W.: Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. *SIAM Rev.* **41**(2), 303–332 (1999)

17. Williams, L., Meneely, A., Shipley, G.: Protection poker: the new software security “game”. *IEEE Secur. Priv.* **8**(3), 14–20 (2010)
18. Zhang, L., Miranskyy, A., Rjaibi, W.: Quantum advantage and the Y2K bug: a comparison. *IEEE Softw.* **38**(2), 80–87 (2021)






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.





Empirical Investigation of Quantum Computing on Solving Complex Problems

Shahid Hussain¹ , Yuba Neupane¹, Wen-Li Wang¹ , Naseem Ibrahim¹ ,
Saif Ur Rehman Khan² , and Asif Kareem³ 

¹ Department of Computer Science and Software Engineering, School of Engineering,
Penn State Behrend, Erie, USA

{shussain, wxw18, nii1}@psu.edu

² Department of Computing, Shifa Tameer-E-Millat University (STMU),
Islamabad 45550, Pakistan

saif_rehman.ssc@stmu.edu.pk

³ Advocate Health, Downers Grove, IL, USA

Abstract. Context: The rules of Quantum Mechanics have been exploited through Quantum Computing (QC) to solve specific problems and process information in expeditious ways as compared to Conventional Computing (CC) such as factoring integers. **Problem:** With the alluring computation capability of QC, it is still important to assess the implications and limitations of QC in solving a variety of computationally demanding problems. **Method:** In this regard, an empirical study was conducted to assess the efficacy of QC in terms of solving certain complex problems by keeping a tradeoff between the execution time and problem size. An analysis was performed based on the widely used Shor's algorithms and the efficacy of QC as compared to CC was reported. **Results:** The outcomes show that QC has the potential to exponentially speed up the identification of a solution to certain polynomial problems that are intractable for CC. However, further research is needed to fully understand the potential and limitations of QC for Non-Polynomial (NP) complete problems.

Keywords: Quantum Computing · Conventional Computing · Shor's Algorithm · Complexity

1 Introduction

Quantum Computing (QC) is a revolutionary technology that exploits the rules of Quantum Mechanics to speedily solve specific problems and process information in ways that are not possible with classic Conventional Computing (CC). In CC, computer scientists are interested in solving optimization and decision problems within polynomial and/or non-polynomial time limit. This is very challenging for NP-complete problems for the problem size can grow exponentially. In contrast to CC for having one state to be on or off at a time, QC can deal with multiple states at the same time, contributing to its high-speed computation performance. Hence, computer science and software engineering communities have endeavored to leverage the discriminative power of QC to

find solutions for difficult optimization and decision problems by possibly maintaining a polynomial time even after a significant growth in problem size. One of such problems is to factor numbers into primes and determine the answer in a reasonable time, especially when the number of digits grows exponentially [1]. Factoring integers is a complex problem that has been widely studied in number theory and cryptography. However, the conventional computers have struggled to factor large integers within a polynomial time until the introduction of Shor's algorithm [2]. The algorithm is a QC approach developed by American mathematician Peter Shor in 1994 to solve this hard problem. Shor's algorithm is also recognized as a strong tool in the cryptographic and code-cracking domain to crack cryptographic algorithms that are frequently used for online communication and trade. This is because many encryption algorithms, including the RSA, are developed based on the challenge of factoring huge integers. The Shor's method can factor large integers significantly faster and more quickly than traditional algorithms, thus defeating the RSA encryption and many other encryption schemes [3].

QC intrigues the community to perform complex tasks by exploiting the principles of quantum mechanics that can account for multiple states simultaneously to gain great performance. Such ability is deemed impossible or impractical to CC with only one state at a time. Today, computer scientists are interested in solving complex problems as quickly as possible using QC even with an exponential growth in the problem size. In [4], Devitt et al. have investigated the practical implementation of Shor's algorithm. In this study, an empirical study is conducted to investigate the impact, ability, and limitation of QC in terms of solving complex problems in polynomial time. A comparative analysis is performed between QC and CC in terms of solving complex problems in polynomial time. Moreover, the experiment explores the effects and benefits of QC in solving less complicated polynomial problems.

2 Related Works

This section gives a summary of the related work, implications, and limitations. In [5], Ugwuishiwu et. al. Investigated the mechanisms of quantum cryptography and compared quantum and classical encryption schemes. In their study, the authors gave a brief overview of quantum computation and explained Shor's algorithm. Moreover, they demonstrated the power of QC in terms of how encryption could be accomplished by utilizing the properties of quantum particles and provided examples of the complexities of Shor's algorithm [5].

Quantum computers and Shor's algorithm can pose a threat to today's cryptographic systems. With the prevalence of data breaches, researchers are increasingly interested in finding ways to safeguard data security using quantum cryptography [4].

In addition to the benefits of QC, Aaronson [6] also identified certain limitations. The investigation showed that quantum computers could be extremely efficient at some specific tasks, where the computation abilities outperformed current computers moderately for most problems. This realization indicates a potential to lead to the discovery of new fundamental physical principles. The concept of a "magic computer" capable of quickly solving NP- complete problems, could change the world, as it could be used to find patterns in large datasets such as stock market data or brain activity recordings

[6]. Nevertheless, the author also claimed that quantum computers are not an all-purpose device. They are best suited for specific types of computations, such as those that involve large amounts of data and require high-speed processing. These are known as quantum advantage tasks, e.g., quantum simulation and quantum cryptography [5, 6].

In [7], Nene and Upadhyay scrutinized a well-known and widely used encryption-based RSA algorithm and investigated the difficulty of factoring large integers within polynomial time. The authors described the capability of Shor’s quantum algorithm in terms of breaking RSA encryption in a reasonable time. The authors further presented a systematic approach to factoring integers using Shor’s quantum algorithm on a classical computer through simulation. The results were verified theoretically and concluded a need for further empirical investigations [8–10].

Thomas et al. [12] tried to leverage and scale Shor’s algorithm in their study on Ion-trap quantum computers (a particular kind of quantum computer). The authors applied the algorithm with the use and manipulation of seven qubits and four “cache qubits”. They factored the number 15 through extended arithmetic operations and modular multipliers. With a degree of confidence of more than 99%, the algorithm was able to provide the proper factors. This is a crucial milestone towards the creation of a scalable quantum computer and the actual use of quantum algorithms [13, 14].

3 Proposed Method

This study empirically assesses the efficacy of QC in solving complex problems and reason the tradeoff between execution time and problem size. The layout of the proposed method is shown in Fig. 1, and the constituent components are described as follows.

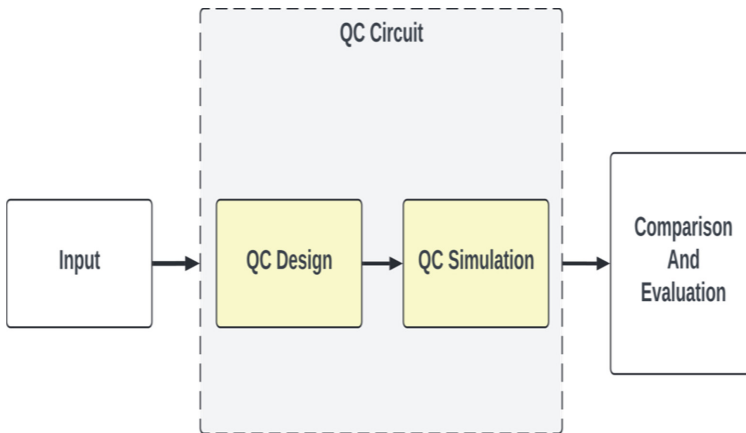


Fig. 1. Workflow of the proposed method

3.1 Input

This component represents the input for a chosen algorithm. Take the Shor’s algorithm for example. The input is an integer value, whose prime factor is expected to be found.

3.2 QC-Circuit

The QC-Circuit in the proposed method is functional in two steps as follows.

3.2.1 Quantum Circuit Design

The layout of the QC-circuit design is shown in Fig. 2. The exponent register is first placed into an equal superposition of states using Hadamard gates. This is a vital step in making sure the adopted quantum algorithm can fully benefit from quantum physics' features. The final qubit in the target register is then subjected to the application of a phase kickback gate. This gate is used to help make sure the algorithm can run smoothly and produce the most accurate result possible.

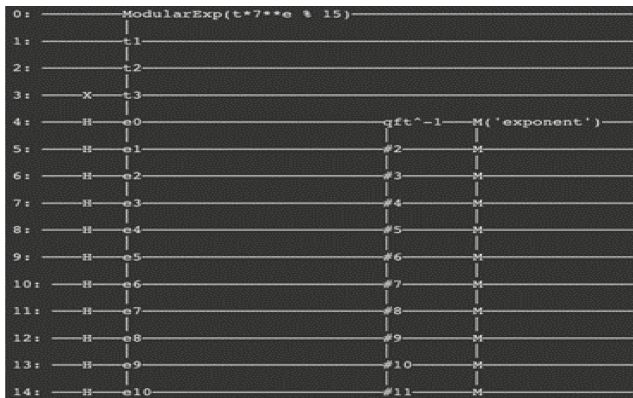


Fig. 2. Quantum Circuit design for input value of 15 in Shor's Algorithm

Afterwards, a modular exponential gate is used to perform the series of controlled unitary units required for phase estimation. This gate, which is an essential part of the process, enables us to precisely determine the phase of the quantum state that is dealt with. Once finished, the next step will apply the inverse quantum Fourier transform to the exponent register. This phase is crucial because it enables us to get pertinent data about the qubits' current state from the register and ensures that the final output is as precise as feasible.

To obtain the outcome, the exponent register will be measured and the data from the register's qubits in this last phase of the procedure can be retrieved. The output of the algorithm is the result of this measurement, and it may be utilized to carry out different computing operations.

3.2.2 Quantum Circuit Design

For simulation of the proposed QC-design, the capabilities of Google's Cirq was leveraged. Using Google's Cirq, the Quantum Virtual Machine enables us to operate and test quantum circuits on simulated hardware that replicates the limitations and noise behavior of real quantum devices. Before placing our quantum algorithms to use on the actual

quantum hardware, the simulator of the virtual machine provides an economical way to test and debug the algorithms.

3.3 Comparison and Evaluation

Google's open-source quantum computing framework, Cirq, provides an accessible platform for researchers and developers to experiment with Shor's algorithm and test it with different integer inputs. With the help of Cirq, we utilized the already implemented Shor's algorithm to run on a quantum simulator by Google, to test the performance of the algorithm in factoring integers with different numbers of digits.

For comparison, the general number field sieve (GNFS) algorithm [15] in number theory was also applied to integers with different numbers of digits because this traditional computing approach can factor integers within a much more reasonable amount of time than the brute-force method documented on Cirq with demo code. However, it's still not as efficient as the quantum algorithm. Our evaluation is based on the size of the input integer and the time required to locate the prime factors. These two are considered as the major criteria for assessing the QC performance in solving the integer factoring problems in polynomial time. The size of the input integer will be determined by the number of digits, which represents how lengthy the number is [9]. The time it takes to locate the prime factors will be measured in seconds and used to reflect the computational efficiency of the algorithm.

The success of comparing the performance of QC with that of the traditional computing approach through the input size and time as two key metrics will enable us to study the influence of further issues on QC performance and identify other potential barriers to its wider adoption involving more problems. The statistics can then provide straightforward and objective results to assess QC's performance.

4 Experimental Procedure

The following steps present our experimental procedure of the proposed study to run the Shor's algorithm.

Step-1: Initializing an input and output qubit register in the quantum computer is the first step, with n and n_0 qubits, respectively, assigned to the state $|\psi_0\rangle = |0\dots 0\rangle_n |0\dots 0\rangle_{n_0}$.

Step-2: The next step is to prepare a superposition by applying q Hadamard gates, where $q = 2n$.

Step-3: The output register of state $|\psi_2\rangle$ is measured. The result of that measurement is then discarded and put into the input register of state $|\psi_3\rangle$.

Step-4: The result y of the input register of state $|\psi_4\rangle$ is measured. This value helps determine the continued fraction representation of y/q . Finally, each convergent result is tested in order and reduced to their lowest terms.

5 Result and Discussion

The result of the proposed method is displayed in Table 1 and Fig. 3. In the table, it shows the measured values of QC and CC in terms of keeping a tradeoff between problem size and polynomial time. Figure 3 on the other hand gives a visual representation of the same results for easy comparison between QC and CC using a bar chart.

The classical computer can take an exponential amount of time to calculate the factors as the number of digits in the integers becomes larger. In contrast, the quantum computer spends about the same amount of time to factor the integers, regardless of the number of digits. The time gap between the quantum and conventional computers becomes more and more significant as the problem size increases. For example, according to the results of Table 1 for the case of 231273, the quantum computer takes less than one second, but the conventional computer takes more than five seconds.

Table 1. Time analysis for factoring using Quantum and Classical computing approaches

Tests	Input	Result	Quantum Computer		Classical Computer	
			Time(s)	Time(ms)	Time(s)	Time(ms)
1	4	[2,2]	0.00099212	0.999212	0.00097537	0.97537
2	6	[2,3]	0.001001596	1.001596	0.001053333	1.053333
3	8	[2,4]	0.000996828	0.996828	0.000994921	0.994921
4	9	[3,3]	0.001024961	1.024961	0.001020432	1.020432
5	10	[5,2]	0.001003265	1.003265	0.000999928	0.999928
6	12	[2,6]	0.000989676	0.989676	0.000981808	0.981808
7	14	[2,7]	0.001004457	1.004457	0.001044273	1.044273
8	15	[3,5]	0.001005457	1.005457	0.001003027	1.003027
9	314	[2,157]	0.001009226	1.009226	0.001000166	1.000166
10	529	[23,23]	0.001008749	1.008749	0.001025677	1.025677
11	1011	[3,337]	0.001010418	1.010418	0.002021385	2.021385
12	23127	[3,7709]	0.000991106	0.991106	0.002010656	2.010656
13	231273	[21,11013]	0.000995636	0.995636	0.005043123	5.043123

Finding 1:

With the increase in problem size, the efficacy of QC over the CC becomes more evident.

The results of Fig. 3 conclude our first finding that quantum computer can be exponentially quicker than the classical computer in solving the proposed algorithm (i.e., Shor's algorithm), especially when the problem size increases. However, the advantage of QC over CC diminishes when the problem size is rather small, adhered to [11].

Finding 2:

As compared to CC, the efficacy of QC could be significantly improved when job is accomplished with a large input size.

It is crucial to remember that the performance of a quantum computer is dependent on a variety of parameters, including the number of qubits. Factoring big numbers is simply one of the many polynomial problems that quantum computers can perform faster

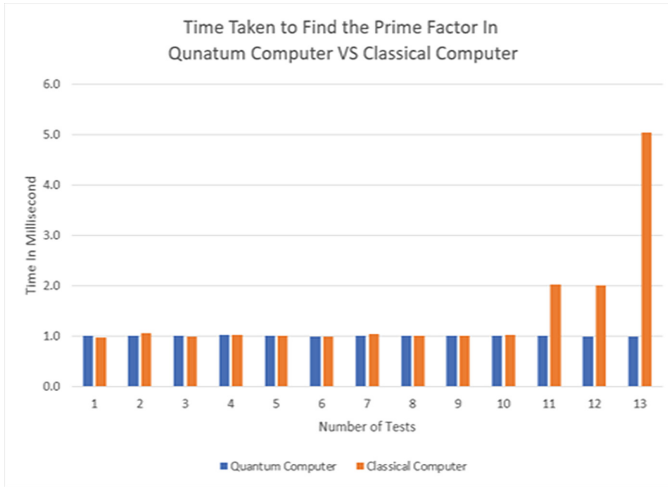


Fig. 3. Comparative analysis of QC and CC for Shor's algorithm

than classical computers. One of the significances is that quantum computers can take a rather constant time, as shown in Fig. 3, to find solutions for the increasing problem sizes. This on the other hand indicates our second finding that QC can process a large input size of data remarkably faster than CC as well. Nonetheless, the two findings do not imply that quantum computers are superior to classical computers in all activities.

6 Implications to Research Community

The aim of the proposed study is to empirically investigate the implications of QC to solve complex problems in terms of polynomial time and its advantage over CC. Through the experimental results of the proposed study, we have realized the following consequences for the research community.

- Quantum computing has the ability to dramatically accelerate the process of factoring big numbers in polynomial time.
- Cryptography and code cracking could be possibly done in polynomial time.
- Rapid resolution for polynomial problems can also have significant effects in other disciplines, including machine learning, optimization, and simulation.
- Researchers might employ quantum computers to more accurately and efficiently model physical systems and tackle challenging optimization issues.
- To secure sensitive information, the improvement to many existing security methods could rely on QC.
- The speed with which complex or hard problems may be solved also suggests that other issues, like as the Traveling Salesman Problem or other kinds of search tasks, which are not known to be time-consuming on classical computers, can be solved using quantum computers.

In short, the speed of quantum computers in rapid problem-solving has the potential to revolutionize numerous industries and offer new opportunities for research and technological development.

7 Conclusion

The experimental results have indicated the efficacy of Quantum Computing (QC) over Classical Computing (CC) to solve complex problems such as optimization and decision problems in terms of polynomial time. This proposed study was conducted to investigate the QC claim regarding its speed of solving complex problems and efficacy over CC. The well-known and widely used Shor's algorithm was exploited, and the capabilities of Google Cirq were leveraged to design and simulate the QC circuit for the algorithm. This empirical study successfully measures the performance of QC and CC and performs a comparative analysis. The conclusions of this work are; 1) With a small problem size, the performance of QC shows no superiority to CC for Shor's algorithm, 2) QC starts to outperform CC when the problem size or input size grows large, 3) Cryptography and code cracking application could rely on QC for its ability to solve complex problems more quickly, and 4) Research community can rely on QC to solve NP-complete problems in a much greater performance, such as Knapsack, Hamiltonian path and Travelling salesman problems.

References

1. Rietsche, R., et al.: Quantum computing. *Electron. Mark.* 1–12 (2022). <https://doi.org/10.1007/s12525-022-00570-y>
2. Voorhoeve, D.: Superposition and entanglement. Quantum Inspire. <https://www.quantum-inspire.com/kbase/superposition-and-entanglement/>
3. Knill, E.: Quantum computing. *Nature* **463**(7280), 441–443 (2010)
4. Devitt, S.J., Fowler, A.G., Hollenberg, L.C.: Investigating the practical implementation of Shor's algorithm. In: *SPIE Proceedings* (2005)
5. Ugwuishiwu, C.H., Orji, U.E., Ugwu, C.I., Asogwa, C.N.: An overview of quantum cryptography and Shor's algorithm. *Int. J. Adv. Trends Comput. Sci. Eng.* **9**(5), 7487–7495 (2020)
6. Aaronson, S.: The limits of quantum. *Sci. Am.* **298**(3), 62–69 (2008). JSTOR. <http://www.jstor.org/stable/26000518>
7. Nene, M., Upadhyay, G.: Shor's algorithm for quantum factoring. In: Choudhary, R., Mandal, J., Auluck, N., Nagarajaram, H. (eds.) *Advanced Computing and Communication Technologies*, pp. 325–331. Springer, Singapore (2016). https://doi.org/10.1007/978-981-10-1023-1_33
8. Google: Shor's algorithm: Cirq. Google Quantum AI. <https://quantumai.google/cirq/experiments/shor>
9. Shor, P.W.: Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. *SIAM J. Comput.* **26**(5), 1484–1509 (1997)
10. Shor, P.W.: Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. *Quantum Phys.* 20–22 (1996)
11. Artur, E., Jozsa, R.: Shor's quantum algorithm for factorizing numbers. *Rev. Mod. Phys.* **68**, 733–753 (1996)

12. Thomas, M., et al.: Realization of a scalable Shor algorithm. *Science* **351**, 1068–1070 (2016)
13. Eleanor, R., Polak, W.: An introduction to quantum computing for non-physicists. *ACM Comput. Surv.* **32**, 300–335 (2000)
14. Li, W., et al.: An image classification algorithm based on hybrid quantum classical convolutional neural network. *J. Quantum Eng.* (2022)
15. Crandall, R., Pomerance, C.: *Prime Numbers: A Computational Perspective*, 2nd edn. Springer, New York (2001). Section 6.2: Number field sieve, pp. 278–301. <https://doi.org/10.1007/978-1-4684-9316-0>

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

Abrahamsson, Pekka 182, 201
Alhammad, Manal 12
Aludhilu, Hilma 116
Alves, Isaque 51
Alves, Samara 51
Anttila, Tatu 201
Attal, Mohammad Idris 192

B

Berger, Christian 173
Bogner, Justus 39

C

Cabrero-Daniel, Beatriz 173
Copei, Sebastian 31
Cordeiro, Renato 51

E

Eckstein, Jutta 3
Eißfeldt, Daniela 21

F

Fraser, Steven D. 63
Fritzsich, Jonas 39

G

Goldman, Alfredo 51

H

Hakala, Jyri 201
Haug, Markus 39
Heimann, Christian 21
Holyer, Steve 3
Hubner-Benz, Sylvia 192
Hussain, Shahid 222
Hyrynsalmi, Sami 125

I

Ibrahim, Naseem 222

K

Kareem, Asif 222
Keller, Alexandra 107
Ketola, Juulia 201
Khan, Saif Ur Rehman 222
Khanna, Dron 87
Kilamo, Terhi 182
Knappe, Anna 201

L

Lahtinen, Daniel 201
Lasi, Heiner 107
Liu, Zixuan 75
Liukko, Väinö 201

M

Mancl, Dennis 63
Marchesi, Lodovica 213
Marchesi, Michele 213
Moe, Nils Brede 149, 161
Moreno, Ana 12
Moroz, Bogdan 125

N

Neupane, Yuba 222
Nurminen, Mikko 182

O

Olsen, John Olav 161

P

Paasivaara, Maria 139
Poranen, Timo 201
Poth, Alexander 21

R

Rafiq, Usman 97, 192
Rantanen, Petri 182

Ritala, Topi-Matti 201
Ronanki, Krishna 173

S

Saari, Mika 182
Saltan, Andrey 125
Schreiter, Maximilian 31
Setälä, Manu 201
Smite, Darja 149
Sporseem, Tor 75
Stray, Viktoria 75, 161
Sutinen, Erkki 116
Systä, Kari 182

T

Tonelli, Roberto 213

W

Wagner, Stefan 39
Wang, Wen-Li 222
Wang, Xiaofeng 87, 97, 139, 192
Waschk, Stefan 21
Werling, Maximilian 107

Z

Zaina, Luciana 97
Zimmermann, Alfred 39
Zündorf, Albert 31