Peggy Gregory
Philippe Kruchten (Eds.)

LNBIP 426

Agile Processes in Software Engineering and Extreme Programming – Workshops

XP 2021 Workshops
Virtual Event, June 14–18, 2021
Revised Selected Papers

Springer

Lecture Notes in Business Information Processing 426

More information about this series at http://www.springer.com/series/7911

Peggy Gregory · Philippe Kruchten (Eds.)

Agile Processes in Software Engineering and Extreme Programming – Workshops

XP 2021 Workshops
Virtual Event, June 14–18, 2021
Revised Selected Papers

Springer

Editors
Peggy Gregory
University of Central Lancashire
Preston, UK

Philippe Kruchten
University of British Columbia
Vancouver, BC, Canada

ISSN 1865-1348 ISSN 1865-1356 (electronic)
Lecture Notes in Business Information Processing
ISBN 978-3-030-88582-3 ISBN 978-3-030-88583-0 (eBook)
https://doi.org/10.1007/978-3-030-88583-0

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

Preface

This volume contains papers from the research workshops at XP 2021, the 22nd International Conference on Agile Software Development, held online during June 14–18, 2021.

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 2021 research papers were published in the conference proceedings (LNBIP, volume 419). This companion volume, published after the conference, contains selected revised workshop papers and workshop summaries.

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 2021, the following five workshops took place:

- 4th International Workshop on Software-Intensive Business
- 9th International Workshop on Large-Scale Agile Development
- 3rd International Workshop on Agile Transformation
- 2nd International Workshop on Agility with Microservices Programming
- 1st International Workshop on Agile Sustainability

In addition to the workshop papers and summaries, these post conference proceedings include abstracts from poster presentations. Finally, we include a summary of the two panel discussions.

We would like to extend our sincere thanks to all the people who contributed to XP 2021: 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.

July 2021

Peggy Gregory
Philippe Kruchten

Organization

Conference Chair

Peggy Gregory	University of Central Lancashire, UK

Workshop Co-chairs

Ademar Aguiar	University of Porto, Portugal
Eduardo Guerra	Free University of Bolzen-Bolzano, Italy

Research Workshops Co-chairs

4th International Workshop on Software-Intensive Business

Karl Werder	University of Cologne, Germany
Sami Hyrynsalmi	Lappeenranta University of Technology, Finland
Xiaofeng Wang	Free University of Bozen-Bolzano, Italy

9th International Workshop on Large-Scale Agile Development

Abdallah Salameh	Bambora AB, Sweden
Julian Bass	University of Salford, UK

3rd International Workshop on Agile Transformation

Diane Strode	Whitireia Polytechnic, New Zealand
Leonor Barroca	The Open University, UK
Marius Mikalsen	SINTEF, Norway

2nd International Workshop on Agility with Microservices Programming

Filipe Figueiredo Correia	University of Porto, Portugal
Florian Rademacher	Dortmund University of Applied Sciences and Arts, Germany
Rebecca Wirfs-Brock	Wirfs-Brock Associates, USA
Blagovesta Kostova	Swiss Federal Institute of Technology, Switzerland

1st International Workshop on Agile Sustainability

Coral Calero	Universidad de Castilla–La Mancha, Spain
Juan Garbajosa	Universidad Politécnica de Madrid, Spain
Jennifer Pérez	Universidad Politécnica de Madrid, Spain
Agustin Yagüe	Universidad Politécnica de Madrid, Spain

Workshop Program Committees Members

Ville Alkkiomäki	F-Secure Corporation, Finland
Idun Backer	Storebrand, Norway
Gavina Baralla	University of Cagliari, Italy
Julian Bass	University of Salford, UK
Hubert Baumeister	Technical University of Denmark, Denmark
Stefan Biffl	Vienna University of Technology, Austria
Tingting Bi	Monash University, Australia
Elizabeth Bjarnason	Lund University, Sweden
Finn Olav Bjornson	SINTEF, Norway
Justus Bogner	University of Stuttgart, Germany
Jan Bosch	Chalmers University of Technology, Sweden
Nils Brede Moe	SINTEF, Norway
Sjaak Brinkkemper	Utrecht University, The Netherlands
Kyle Brown	IBM, USA
Christoph Bussler	Google, USA
Sarah Beecham	Lero, Ireland
Hong-Mei Chen	University of Hawaii, USA
Ruzanna Chitchyan	University of Bristol, UK
Torgeir Dingsøyr	Norwegian University of Science and Technology, Norway
Bettina Distel	University of Münster, Germany
Denniz Dönmez	ETH Zürich, Switzerland
Andreas Drechsler	Victoria University of Wellington, New Zealand
Jutta Eckstein	IT communication, Germany
Henry Edison	Lero and National University of Ireland, Galway, Ireland
Christoph Elsner	Siemens AG, Germany
Hendrik Esser	Ericsson, Sweden
Fabian Fagerholm	Aalto University, Finland
Fábio Fagundes Silveira	Federal University of Sao Paulo, Brazil
Michał Gajda	MigaMake, Singapore
Alfredo Goldman	University of São Paulo, Brazil
Paul Grünbacher	Johannes Kepler Universität Linz, Austria
Eduardo Guerra	Free University of Bolzen-Bolzano, Italy
Tomas Gustavsson	Karlstad University, Sweden
Robert Heinrich	Karlsruhe Institute of Technology, Germany
Andy Haxby	Competa, The Netherlands

Georg Herzwurm	University of Stuttgart, Germany
Helena Holmström Olsson	Malmö University, Sweden
Bettina Horlach	Hamburg University, Germany
Sami Hyrynsalmi	LUT University, Finland
Sonja Hyrynsalmi	LUT University, Finland
Slinger Jansen	Utrecht University, The Netherlands
Rick Kazman	Carnegie-Mellon University and University of Hawaii, USA
Hans-Bernd Kittlaus	InnoTivum Consulting, Germany
Eric Knauss	University of Gothenburg, Sweden
Dina Koutsikouri	University of Gothenburg, Sweden
Maarit Laanti	Nitor Delta, Finland
Ville Leppänen	University of Turku, Finland
Alexander Mädche	Karlsruhe Institute of Technology, Germany
Andrey Maglyas	Maglyas Consulting, Russia
Carl Marnewick	University of Johannesburg, South Africa
Santiago Matalonga	University of the West of Scotland, UK
Claudia Melo	International Atomic Energy Agency, UN
Tommi Mikkonen	University of Helsinki, Finland
Suzanne Miller	Deloitte, USA
Sunila Modi	University of Hertfordshire, UK
Parastoo Mohagheghi	Labour and Welfare Administration, Norway
Jürgen Münch	Reutlingen University, Germany
Ingo Müller	Monash University, Australia
Stefan Naumann	Trier University of Applied Sciences, Germany
Anh Nguyen Duc	University College of Southeast Norway, Norway
Alexander Nolte	University of Tartu, Estonia
John Noll	University of Hertfordshire, UK
Cesare Pautasso	University of Lugano, Switzerland
Marco Peressotti	University of Southern Denmark, Denmark
Andrea Pinna	University of Cagliari, Italy
Alexander Poth	Volkswagen, Germany
Ken Power	Independent Consultant, Ireland
Jan Pries-Heje	Roskilde University, Denmark
Rafael Prikladnicki	PUCRS, Brazil
Jurka Rahikkala	Vaadin Ltd, Finland
Scarlet Rahy	University of Salford, UK
Bala Ramesh	Georgia State University, USA
Seb Rose	Claysnow, UK
Guenther Ruhe	University of Calgary, Canada
Alceste Scalas	Aston University, UK
Helen Sharp	The Open University, UK
Abdallah Salameh	Bambora AB, Sweden
Kari Smolander	LUT University, Finland
Jacopo Soldani	University of Pisa, Italy

Jonas Sorgalla	Dortmund University of Applied Science and Arts, Germany
Nuno Santos	Polytechnic Institute of Viana do Castelo and ALGORITMI Research Center, Portugal
Viktoria Stray	University of Oslo, Norway
Katie Taylor	Agile Business Consortium, UK
Roberto Tonelli	University of Cagliari, Italy
Pasi Tyrväinen	University of Jyväskylä, Finland
Siffat Ullah Khan	University of Malakand, Pakistan
Stefan Wagner	University of Stuttgart, Germany
Xiaofeng Wang	Free University of Bozen-Bolzano, Italy
Hironori Washizaki	Waseda University, Japan
Hasan Yasar	Carnegie Mellon University, USA
Joseph Yoder	The Refactory, USA
Olaf Zimmermann	University of Applied Sciences of Eastern Switzerland, Switzerland

Poster Track Chairs

Noel Carroll	Lero and National University of Ireland Galway, Ireland
Kashumi Madampe	Monash University, Australia

Publication Chair

Philippe Kruchten	University of British Columbia, Canada

XP Steering Committee

Hubert Baumeister	Technical University of Denmark, Denmark
François Coallier	Ecole de technologie supérieure, Canada
Jutta Eckstein	IT communication, Germany
Steven Fraser	Innoxec, USA
Juan Garbajosa (chair)	Universidad Politécnica de Madrid, Spain
Peggy Gregory	University of Central Lancashire, UK
Ellen Grove	Agile Alliance, USA
Casper Lassenius	Aalto University, Finland
Michele Marchesi	University of Cagliari, Italy
Maria Paasivaara	IT University of Copenhagen, Denmark, and Aalto University, Finland
Viktoria Stray	University of Oslo, Norway
Xiaofeng Wang	Free University of Bozen-Bolzano, Italy

Sponsoring Organization

Agile Alliance	USA

Contents

3rd International Workshop on Agile Transformation

Agile Transformation at Scale: A Tertiary Study 3
Suddhasvatta Das and Kevin Gary

Institutional Logics in Large-Scale Agile Software Development Transformations 12
Tomas Gustavsson

9th International Workshop on Large-Scale Agile Development

Innovation in Large-Scale Agile - Benefits and Challenges of Hackathons When Hacking from Home 23
Rasmus Ulfsnes, Viktoria Stray, Nils Brede Moe, and Darja Šmite

Impacts of COVID-19 Pandemic for Software Development in Nordic Companies – Agility Helps to Respond 33
Petri Kettunen, Tomas Gustavsson, Maarit Laanti, Andreas Tjernsten, Tommi Mikkonen, and Tomi Männistö

The EFIS Framework for Leveraging Agile Organizations Within Large Enterprises 42
Alexander Poth, Mario Kottke, Christian Heimann, and Andreas Riel

Managing Dependencies in Large-Scale Agile 52
Henrik Vedal, Viktoria Stray, Marthe Berntzen, and Nils Brede Moe

Summary of First International Workshop on Agile Sustainability

How Collective Intelligence Can Gear Agility with Sustainability 69
Juan Ochoa-Zambrano

Summary of 4th International Workshop on Software-intensive Business

SaaS Pricing Practices Typology: A Case Study 87
Andrey Saltan and Kari Smolander

Is Your Software Ecosystem in Danger? Preventing Ecosystem Death Through Lessons in Ecosystem Health 96
Robert Evertse, Abel Lencz, Tea Šinik, Slinger Jansen, and Lamia Soussi

Genesis of a Wood Harvesting B2B Software Platform 106
Jaakko Vuolasto and Kari Smolander

Towards a Taxonomy of Impact Factors for Digital Platform Pricing. 115
Virginia Springer and Dimitri Petrik

Assessing the Health of the Dark Web:: An Analysis of Dark Web Open Source Software Projects 125
Samuel Onyango, Emilie Steenvoorden, Joram Scholten, and Slinger Jansen

Using Guilds to Foster Internal Startups in Large Organizations: A Case Study 135
Tor Sporsem, Anastasiia Tkalich, Nils Brede Moe, Marius Mikalsen, and Nina Rygh

Employee-Driven Innovation to Fuel Internal Software Startups: Preliminary Findings 145
Anastasiia Tkalich, Nils Brede Moe, and Tor Sporsem

Towards a Framework to Guide the Creation of Development Practices for Software Startups. 155
Jorge Melegati

2nd Workshop on Agility with Micro Service Programming

Towards Integrating Blockchains with Microservice Architecture Using Model-Driven Engineering 167
Simon Trebbau, Philip Wizenty, and Sabine Sachweh

A Service Mesh for Collaboration Between Geo-Distributed Services: The Replication Case. 176
Marie Delavergne, Ronan-Alexandre Cherrueau, and Adrien Lebre

Implementation of a Microservice-Based Certification Platform. 186
Sebastian Copei, Manuel Wickert, and Albert Zündorf

Poster Presentations

Multiple Roles of Middle Managers in Agile Project Governance: An Activity Theory Perspective 195
Maduka C. Uwadi

Cherry Picking - Agile Software Development Teams Applying Design Thinking Tools 201
Franziska Dobrigkeit, Christoph Matthies, Philipp Pajak, and Ralf Teusner

From Project to Product 207
Matthew Philip and Yoan Thirion

Panels

The Stories We Tell: Experience, Research, or Patterns? 215
Dennis Mancl and Steven D. Fraser

The Future of Software Engineering: Where Will Machine Learning, Agile, and Virtualization Take Us Next? 222
Dennis Mancl and Steven D. Fraser

Author Index 231

3rd International Workshop on Agile Transformation

Agile Transformation at Scale: A Tertiary Study

Suddhasvatta Das and Kevin Gary

Arizona State University, Tempe, AZ 85287, USA
{sdas76,kgary}@asu.edu

Abstract. Due to the fast-paced nature of the software industry and the success of small agile projects, researchers and practitioners are interested in scaling agile processes to larger projects. Agile software development (ASD) has been growing in popularity for over two decades. With the success of small-scale agile transformation, organizations started to focus on scaling agile. There is a scarcity of literature in this field making it harder to find plausible evidence to identify the science behind large scale agile transformation. The objective of this paper is to present a better understanding of the current state of research in the field of scaled agile transformation and explore research gaps. This tertiary study identifies seven relevant peer reviewed studies and reports research findings and future research avenues.

Keywords: Large scale agile transformation · Tertiary study

1 Introduction

Transformation from a traditional software engineering process to an agile process is not well understood. This area of agile transformation is relatively new, and researchers are working on different aspects to gain new understanding. Industry organizations that transformed from waterfall to agile implemented multiple strategies such as altering quality assurance practices [1, 2] and training staff in agile methods [3, 4]. However, organizations must consider additional factors when scaling such transformations.

In this paper, we present a tertiary study in scaled agile transformations. Our focus is on large projects defined by the number of people working on the project team. We are not aware of another tertiary study specific to scaled agile transformation. This study identifies gaps in the literature of scaled agile transformation that will help the community to identify potential research avenues in future.

We could not identify tertiary studies that present a meta-analysis of relevant secondary studies. Additionally, we observed there are not many secondary studies compared with more mature research areas. The goal for this study is to collate instead of synthesize; in this way we hope to identify gaps as opportunities for further study and understanding. As a relatively new area, researchers have been trying to answer different questions related to the scaled agile transformation. The contribution of this paper is to assemble evolving early large-scale agile transformation evidence.

The main goal of this tertiary study is to synthesize the research goal and findings of peer-reviewed secondary studies in scaled agile transformation. Under this goal, we addressed two specific questions:

P. Gregory and P. Kruchten (Eds.): XP 2021 Workshops, LNBIP 426, pp. 3–11, 2021.
https://doi.org/10.1007/978-3-030-88583-0_1

RQ1: What success factors and challenges in scaled agile transforms have been identified by prior secondary studies?
RQ2: What gaps exist in existing studies that should be prioritized by the research community moving forward?

2 Research Methodology

We use [5] and [6] as guidelines for performing this tertiary study. The research methodology was primarily conducted by the first author and reviewed by the second author.

2.1 Search Process

The steps of the search process are shown in Fig. 1 and summarized as follows:

1. Digital libraries from IEEE (29), ACM (12), SpringerLink (35), and ScienceDirect (46) were searched for journal, conference, and workshop papers.
2. Duplicate studies were removed manually.
3. For a tertiary study, we only consider secondary studies for analysis. Specifically, only systematic literature reviews (SLRs) were included beyond this step.
4. The abstracts from step 3 were read. Reviewers searched for keywords and for verification that the study was an SLR on the topic of scaled agile transformation. Manual review was necessary as an initial keyword scan on terms such as 'agile', 'large', 'scale', 'change' OR 'transformations' resulted in too many false positives.
5. The final set of papers was determined by a full-text manual review of the studies from step 4, conducted by both authors, focusing on research questions and paper quality. Paper quality review may be subjective [5]; we considered the publication venue, date of publication, citations of the paper, and citation indices of the venue and authors. Further, we also considered whether enough primary studies were included in the secondary study to validate claims of significance. Disagreements between the authors on inclusion or exclusion were resolved through discussion.

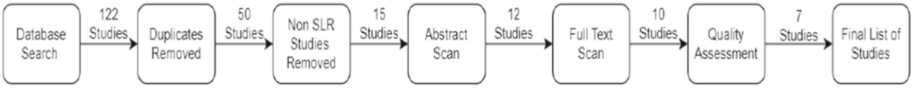


Fig. 1. Search process for identifying papers to include in the tertiary study.

2.2 Summary of Included Studies

Agile adoption has rapidly increased over the past two decades. A few preliminary studies and calls for more research in scaled agile transformation started appearing in the 2000s as practitioners in large organizations are moved towards large-scale agile

Table 1. Final set of papers included in the study.

Study	Venue	Num	Year	Research Goal
[12]	Journal	52	2016	Challenges/success factors in large scale transformations
[13]	Conference	73	2018	Challenges in large scale agile development
[11]	Journal	9	2018	Challenges and success factors in large scale agile
[14]	Journal	19	2019	Supporting Software Product Line engineering in large-scale agile transformation
[15]	Conference	43	2019	Challenges of scaling agile software projects
[16]	Conference	51	2017	Review of Success Factors for Scaling Agile in Global Software Dev Environments
[17]	Journal	20	2015	Scaling approaches, frameworks, and limitations

adoption (cf. [7–10]). Scholarly study of this phenomenon largely started appearing since 2015. Given the relatively small number of studies that met our inclusion criteria, we summarize these papers here.

Dikert, Paasivaara & Lassenius [12] present an extensive secondary study of 52 primary studies on the challenges and success factors of agile transformation. The study reported 35 challenges classified into 9 categories and 29 success factors classified into 11 categories. As per the authors, the most important success factors were management support, choosing the right agile model, mindset, and alignment with the organization's value. The challenges included resistance to change, lack of training, and misunderstanding agile. This study is very influential and a widely referenced work, however, all of the primary studies used for evidence were from 2010 and earlier, and almost all describe transformation from a waterfall-like process to an agile process.

Uludag et al. [13] conducted a structured literature review of 73 papers related to the challenges of scaling agile from a stakeholder perspective. The study reported 79 challenges in 11 categories. The top 3 challenges were coordinating multiple teams working on the same project, considering dependencies in integration, and coordination among geographically distributed teams. This recent study reports new challenges in scaled agile transformations. Notably, it claims the majority of challenges in large-scale agile development (38 of 79) still exist and are "typical". The stakeholder perspective focuses on development team roles, with only a few higher-level roles included.

An action research approach to a meta-review was taken by Kalenda, M., Hyna, P., & Rossi, B [11]. The authors identified 8 common features of scaling frameworks SAFe and LeSS [10] and used this to drive a focused literature review. 12 papers were selected, 10 of which mapped to at least one of the 8 common features, and then described challenges and success factors of each paper. The paper is influential due to its recency and use of scaling frameworks (whose awareness and adoption are becoming more prevalent in industry) as an organizing principle. However, the authors acknowledge this approach is not comprehensive, so there may be more evidence in the literature not included in the study. We also wonder if the approach of identifying common practices between

SAFe and LeSS is appropriate, due to scaling agile not being a prescriptive formulaic process, and also because multiple frameworks were excluded (notably DaD [10] and SoS [10]). Nevertheless, this paper is highly influential and an ambitious action research for understanding scaled agile transformation.

Klünder et al. [14] answer 4 questions on large-scale agile transformations. On the first question *'Does any transformation model for large companies exist that-in particular-preserves already existing SPLs?'* The authors state that no model can be used to transform large organizations into agile. The second question *'What preconditions should be met before starting transformation toward in a large company?'* indicates success factors reported by studies which are the same as [11], [12], and [17] such as management commitment, training, knowledge, and one additional precondition that is risk planning. Question 3 '*What tasks are recommended to be fulfilled during the agile transformation on development team and management level?'* reports that the distribution of tasks and setting up the infrastructure are key steps that need to be done by development teams and management during the transformation phase. Question 4 *'What tasks are required on organizational level to finalize the transformation in a large company?*' reports that management should start the transformation with a pilot team so they can get feedback to improve the pilot team's transformation and also other teams.

A different set of challenges related to scaling agile software development has been reported by Ozkan & Tarhan [15]. The study reports physical dependencies, fragmentation, feudalism, narrow focus on product, construction, and bottlenecks from one to many. This paper is relatively new, but reported some challenges that we could not find any other studies we analyzed.

Shameem et al. [16] reports success factors for large scaled agile projects. The authors report a set of 15 success factors grouped into six categories. The paper also classified these into two major categories, client and vendor. This provides a broader picture of agile processes and factors related to its respective success factors.

Saeeda et al. [17] reports that 24% of the research states that documentation is one of the limitations of agile, 22% reported time period issues and 14% talks about budget overflow. It also reports that 33% of studies report communication as a challenge and 25% report distributed teams as a challenge. The authors also report that researchers are working to find the limitations of agile scalability and its remedial ways.

2.3 Data Extraction

We extracted detailed information from the 7 studies in Table 1 including research goals and questions, findings, discussion, and limitations and reviewed this information for our analysis. Our analysis focused on the success factors and/or challenges in scaled agile transformation presented in each paper, though we note this was not always the primary focus of every study. For example, study [14] presents literature review findings from primary studies a bit differently; this paper identifies *preconditions* and *tasks* from the primary studies that exist for large-scale agile transformation to be successful. We mapped preconditions and tasks to success factors to facilitate analysis of these studies. Tables 2 and 3 below shows 23 different challenges and 22 success factors reported by the 7 studies from Table 1.

Table 2. Challenges in Scaled Agile Transformations Reported by Prior Studies.

Challenges	Description
Resistance to change (CH1)	Employees not willing to work in a new way
Coordination/Communication (CH2)	Teams not working together. Stakeholders not communicating leading to errors
Requirements engineering (CH3)	Vague/incorrect requirements
Quality assurance (CH4)	Quality of the S/W compromised
Integration (nonfunctional requirements) (CH5)	Difficulty making everything work together
Management (CH6)	Non supporting leaders
Tech debt (CH7)	Solution not serving the bigger creates issues
Difficult to implement (CH8)	Difficulty in executing agile
Training (CH9)	Stakeholders have wrong or not enough knowledge about agile
Lack of commitment (CH10)	Stakeholders not committed to a new way of working
Too much workload (CH11)	Employees end up working more than required
Distributed team/ Physical dependencies (CH12)	Teams in multiple geographic location
Measuring progress (CH13)	Difficulty in keeping track of the tasks
Different approaches among teams (CH14)	Different ways of interpreting agile
Lack of investment (CH15)	No budget to educate stakeholders in agile
Fragmentation feudalism (CH16)	Teams relying on directions from others
Short & static event (CH17)	Not able to work in a short amount of time
Narrow focus on products (CH18)	Focusing too much on the S/W Dev rather than the solution to the problem
Narrow focus on construction (CH19)	Focusing too much on the S/W construction rather than the solution to the problem
Bottle neck (one: many relations) (CH20)	Difficulty when in changing product backlog when multiple teams work on one product
Documentation (CH21)	People either doing over or no documentation
Budget overflow (CH22)	Project costs exceeds budget
Human Resources (CH23)	Problems related to HR rules

We manually combined similar ideas with different verbiage into one for the purpose of this study. For example, ‘Change Resistance’ from [12] and ‘Dealing with doubts in people about changes’ [13] have been combined as ‘Resistance to change’.

Table 3. Success Factors in Scaled Agile Transformations Reported by Prior Studies

Success Factors	Description
Management support and Leadership (SF1)	Good support from management
Acquire knowledge (SF2)	Learn from previous experiences
Requirement engineering (SF3)	Requirements perfectly done before working
Communication (SF4)	Stakeholders in sync with each other
Self-organizing teams (SF5)	Teams don't rely on anyone to guide them daily
Engaging people in events (SF6)	Platform so people get to know their co-workers
Tools and infrastructure (SF7)	Technologies to support agile environment
Customer involvement (SF8)	Customer in the loop from project start to end
Short iteration (SF9)	Keep sprints short
Small team size (SF10)	Involve a smaller number of people
Choosing/customizing agile approach (SF11)	Selecting and tailoring the right agile process
Piloting (SF12)	Start with a one project rather than all
Project visibility (SF13)	Stakeholders having the bigger picture
United views (SF14)	Stakeholders sharing same ideas for the project
Training (SF15)	Stakeholders should be trained in agile
Planning (including risk planning) (SF16)	Plan the project and potential risks
Assessment of the S/W dev process (SF17)	Constantly evaluate & improve the dev process
Budget (SF18)	Keep a track of budget
Distributing tasks (SF19)	Distribute tasks among all members
Continuous feedback (SF20)	Get feedback from stakeholders in all the steps of development
Experienced developers (SF21)	Have senior developers to work efficiently
Motivating developers (SF22)	Keep developers motivated

2.4 Limitations

There are challenges in conducting a tertiary review in a topic as recent and fluid as scaled agile transformation. First, there is a lack of general agreement on what is *scale*. The term can describe the size of an organization, the size of software projects, the breadth of application and system domains, or the range of organizational roles participating in the transformation. Second, the recency of industry adoption and published research presented a challenge both for identifying relevant literature and for applying a systematic process for analysis. Admittedly, we had to soften our inclusion criteria somewhat to identify even the small number of recent studies due to this limitation. Analyzing the papers from a common perspective was also difficult as often the studies focused on different aspects. For example, [13] focused on stakeholder perspectives, [11] started with scaled agile frameworks (SAFe and LeSS), [14] focused on software product line engineering, and [15] focused on a design perspective. Therefore, our identification of common success and/or challenge factors is bounded by the perspectives of the included secondary studies. Finally, this study was conducted by one Ph.D. student as the primary researcher, and a single secondary researcher. In a situation where the research is sparse and there is an above average reliance on subjective interpretation due to the subject matter, a third researcher may have improved the arbitration process.

3 Analysis and Discussion

In this section we present the answers to the research questions.

RQ1: What success factors and challenges in scaled agile transforms have been identified by prior secondary studies?

From Tables 2 and 3 we see that 5 out 7 studies identified challenges while 4 out 7 studies presented success factors in scaled agile transformations. Two studies [11] and [17] reported transformation frameworks and limitations. We could find only one study [14] that suggests organization support is a key factor in scaled agile transformation.

Figure 2 (left) shows the coverage of challenges by different studies as listed in Table 2. In this format, we can see there is not widespread agreement on the challenges, though the few that are agreed upon include coordination, employee mindset, management support, resistance to change and quality assurance.

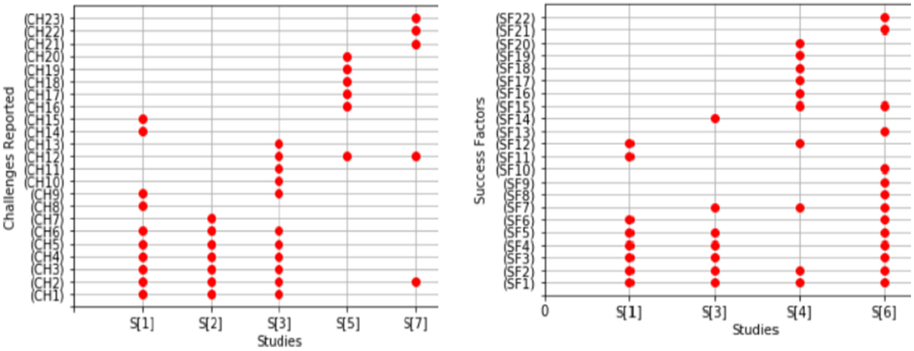


Fig. 2. Challenges (left) and Success Factors (right) reported by previous studies

Figure 2 right shows the coverage of success factors by different studies listed in Table 3. There is somewhat more agreement in factors here as compared to challenges, though again there are still some (such as training, coordination, training and knowledge) that are emphasized in most all, if not all, studies.

RQ2: What gaps exist in current studies that should be prioritized by the research community moving forward?

The answer to this question will shed light on the future research avenues. Scaled agile transformation is a relatively new area so we could only identify a few relevant studies. Most of the studies focus on challenges and success factors of scaled agile transformation. Given the scarcity of literature and the answers from RQ1 these are the research gaps we identified that the community needs to address to move forward.

There are some challenges to scale agile projects that have been reported by a significant number of studies (Table 2). However, there are still many challenges that appeared only in one of the seven studies in Table 1 (CH7, CH8, CH10, CH11, CH13–CH22). Success factors are similar; some were reported by multiple studies (Table 3) while others (SF8–SF11, SF13, SF14, SF16–SF22) were reported by only one study.

In our opinion the identification of challenges and success factors by these studies offer guidance to real-world practitioners and identify areas for future research. Further research is needed to identify common perspectives as more software engineering organizations go from "agile-in-the-small" to "agile-in-the-large" transformation. The ultimate goal is to coalesce understanding into a reference framework for practitioners such a machine learning or statistical model. The goal of the framework would be to help practitioners to make decisions during scaling agile. These are the possible avenues that we think need to be explored.

References

1. Huo, M., Verner, J., Zhu, L., Babar, M.A.: Software quality and agile methods. In: Proceedings of the 28th Annual International Computer Software and Applications Conference, COMPSAC 2004, pp. 520–525. IEEE (2004)
2. Ambler, S.: Quality in an agile world. Softw. Qual. Prof. **7**(4), 34 (2005)
3. Conboy, K., Coyle, S., Wang, X., Pikkarainen, M.: People over process: key people challenges in agile development (2011)
4. Da Silva, F.Q., Santos, A.L., Soares, S., França, A.C.C., Monteiro, C.V., Maciel, F.F.: Six years of systematic literature reviews in software engineering: an updated tertiary study. Inf. Softw. Technol. **53**(9), 899–913 (2011)
5. Kitchenham, B., et al.: Systematic literature reviews in software engineering–a tertiary study. Inf. Softw. Technol. **52**(8), 792–805 (2016)
6. Lindvall, M., et al.: Empirical findings in agile methods. In: Wells, D., Williams, L. (eds) XP/Agile Universe 2002. LNCS, vol. 2418, pp. 197–207. Springer, Heidelberg (2002). https://doi.org/10.1007/3-540-45672-4_19
7. Fry, C., Greene, S.: Large scale agile transformation in an on-demand world. In: Agile 2007 (AGILE 2007), pp. 136–142. IEEE (2007)
8. Beavers, P.A.: Managing a large "Agile" software engineering organization. In: Agile 2007 (AGILE 2007), pp. 296–303. IEEE (2007)
9. Lee, E.C.: Forming to performing: transitioning large-scale project into agile. In: Agile 2008 Conference, pp. 106–111. IEEE (2008)
10. Alqudah, M., Razali, R.: A review of scaling agile methods in large software development. Int. J. Adv. Sci. Eng. Inf. Technol. **6**(6), 828–837 (2016)
11. Kalenda, M., Hyna, P., Rossi, B.: Scaling agile in large organizations: Practices, challenges, and success factors. J. Softw.: Evol. Process **30**(10), e1954 (2018)

Secondary Studies identified for this paper:

12. 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)
13. Uludag, Ö., Kleehaus, M., Caprano, C., Matthes, F.: Identifying and structuring challenges in large-scale agile development based on a structured literature review. In: 2018 IEEE 22nd International Enterprise Distributed Object Computing Conference (EDOC), pp. 191–197. IEEE (2018)
14. Klünder, J.A.C., Hohl, P., Prenner, N., Schneider, K.: Transformation towards agile software product line engineering in large companies: a literature review. J. Softw.: Evol. Process **31**(5), 1–23 (2019)
15. Ozkan, N., Tarhan, A.K.: Investigating causes of scalability challenges in agile software development from a design perspective. In: 2019 1st International Informatics and Software Engineering Conference (UBMYK), pp. 1–6. IEEE (2019)
16. Shameem, M., Kumar, C., Chandra, B., Khan, A.A.: Systematic review of success factors for scaling agile methods in global software development environment: a client-vendor perspective. In: 2017 24th Asia-Pacific Software Engineering Conference Workshops (APSECW), pp. 17–24. IEEE (2017)
17. Saeeda, H., Khalid, H., Ahmed, M., Sameer, A., Arif, F.: Systematic literature review of agile scalability for large scale projects. Int. J. Adv. Comput. Sci. Appl. **6**(9), 63–75 (2015))

Institutional Logics in Large-Scale Agile Software Development Transformations

Tomas Gustavsson(✉)

Karlstad University, 651 88 Karlstad, Sweden
tomas.gustavsson@kau.se

Abstract. Transforming into agile ways of working in large organizations can be performed in different ways. Many organizations choose a defined large-scale agile software development framework but how the transformation is carried out could be based on different sorts of logics. This paper investigates institutional logics at play in large-scale agile transformations. By studying two case organizations, the paper aims at improving our understanding of large-scale transformations by viewing software development as an institution. The findings displays diverse impacts due to two differing institutional logics when transforming into large-scale agile software development by implementing the Scaled Agile Framework. One contribution of this paper is to show the possibilities of using two institutional logics, Agile toolbox logic and Agile rulebook logic, for analyzing impacts of agile transformations.

Keywords: Agile software development · Agile transformation · Large-scale · Institutional logics

1 Introduction

Agile software development methods are increasingly being implemented in larger organizations [1, 2]. More research on large-scale agile ways of working is called for, especially regarding organizational transformation [3, 4]. When transforming an organization into large-scale agile ways of working, roles, routines, practices, and actions are added, changed and adapted. This means that large-scale agile transformations of organizations is more than agile software development. Transformations usually reaches far beyond software development. Often, the purpose of the change is to implement strategic agility in an organization [5]. This entails reducing frictions between autonomous agile teams on one hand and traditional top-down organizational routines with annual planning cycles on the other hand. Since large-scale agile ways of working contains symbols, roles, routines, and actions taken for granted, it could be considered an informal institution [6].

Many organizations choose a predefined large-scale agile framework such as the Scaled Agile Framework (SAFe), Large-Scale Scrum (LeSS), or the Spotify Model [2]. Implementing a large-scale agile framework implies a transformation of roles and routines. Large-scale agile frameworks come with predefined structures and routines,

P. Gregory and P. Kruchten (Eds.): XP 2021 Workshops, LNBIP 426, pp. 12–19, 2021.
https://doi.org/10.1007/978-3-030-88583-0_2

and implementing them within an existing organizational structure is challenging [7]. In addition, large-scale frameworks often require changing the organizational structures.

There are different ways to implement roles and routines based on a framework. When an agile framework is implemented, there is a tendency to measure agile transformation by adherence to that framework [7]. By using this measurement, the framework becomes the rulebook where as much as possible should be implemented (otherwise, we are not "following the rules"). Another way of implementation is by selecting roles and routines based on the challenges in the organization [8]. In this approach, the framework is seen as a toolbox, where parts of the framework are cherry-picked for the organization. Both of these approaches are common in framework implementations, but the impacts on the organization are not much studied. Therefore, the purpose of this paper is to improve our understanding of how differing institutional logics have an impact on large-scale agile transformations.

2 New Institutional Theory

New institutional theory is a loosely coupled body of knowledge accumulated in several streams of research [9] rather than a coherent theory.

Institutional logics is a concept in sociology and organizational studies which focus on how broader belief systems shape the cognition and behavior of individuals [9]. Rather than physical realities driving human action, institutions are enacted through reproduced patterns of activities by individuals.

Friedland and Alford [10] identified several key institutions such as the nuclear family and bureaucratic state, each guided by a distinct institutional logic. A set of goals, values, and prescriptions associated with a specific institution form an institutional logic. Therefore, to be able to understand both individual and organizational behavior, it must be located in an institutional context which both regularizes behavior and, at the same time, provides an opportunity for agency and change. An institutional logic could be described in several dimensions, or elemental categories, such as with a root metaphor and sources of legitimacy, authority and identity [11].

The concept of software development as an institution is not new, as Rowlands [12] presents in his work. Doležel [13] suggested that the software development institution is driven by two disparate institutional logics: Traditional Software Engineering logic and Agile Software Development logic. Attempts have been made to investigate differing logics within agile software development as an institutional logic. Berente et al. [14] studied three agile software development projects and suggested three different institutional logics based on differing contexts.

Another development of institutional logics in agile software development are two dichotomous logics based on differing views, principles and preferences on implementing a large-scale agile method or framework [15]. These logics can be presented based on the differing dimensions, or elemental categories, as Thornton et al. [11] calls them (see Table 1). Elemental categories specifies organizing principles that shape preferences and interests [11].

One of these logics is called *Agile rulebook logic* [15]. The logic means that a proper agile method is chosen and implemented in full as described. Šmite et al. [8] calls this

Table 1. Institutional logics in large-scale agile transformations [15].

Dimension	Agile toolbox logic	Agile rulebook logic
Root Metaphor	Agile routines as tools	Agile routines as rules
Sources of Legitimacy	Unity of improvement needs	Unity in interpretations, standardization
Sources of Authority	Commitment to team decisions	Commitment to method descriptions
Sources of Identity	Attentiveness and responsiveness	Method knowledge
Basis of Attention	Routine inefficiencies	Level of method adoption
Basis of Strategy	Implement, tailor and/or invent routines	Implement the method of choice

the *all-or-nothing attitude*. The method description becomes a rulebook which guides the organization in how to implement roles, practices and routines. The commitment to the method descriptions becomes the source of authority (see Table 1). The basis of strategy for tailoring according to context is therefore based on which method to choose, rather than to tailor the method itself (see Table 1).

Some advocates of this approach are the originators of Scrum, Schwaber and Beedle [16], who argued that agile methods cannot be applied by cherry picking but must be applied in their entirety to achieve the desired effect.

The other logic is called *Agile toolbox logic* which means that roles, routines, and practices are selected and tailored based on challenges in the organization. Šmite et al. [8] calls this the *à la carte approach*. The basis of strategy is to construct a situationally appropriate method out of existing method fragments or innovate based on context needs (see Table 1). The commitment to team decisions on tailoring during transformation becomes the source of authority, rather than method descriptions (see Table 1). Advocates of this logic are, e.g., McBreen [17] and Fitzgerald et al. [18] who argue that parts of the Agile methods can be cherry-picked, ignored or replaced.

3 Research Method

This case study is based on two cases, Case A and Case B, of large-scale agile software development transformations. The case study approach [19] was deemed most suitable for the purpose of the study, as a rich and in-depth understanding of the organizational transformation was sought. Case A is a pilot transformation project where two departments at a large government agency merged into one unit with new roles and routines. The aim of the pilot project was to find out best practices for transforming roles and organizational routines. The next step would then be to use the experience from this transformation to further transform the roles and routines within the whole agency. Case B is a department responsible for the product development of a significant part of a motor vehicle, both the software and the hardware. The department was organized into twenty teams and, after experiencing coordination difficulties between the teams, they decided to start an organizational transformation where new roles and routines were to be implemented.

Semi-structured interviews were the most important source of information but were supplemented with other data collected from participant observations and documents. Several data sources were used for triangulation purposes (see Table 2).

Table 2. Data sources.

Data source	Case A	Case B
Hours of observation	113 h	196 h
Number of interviews	6	14
Hours of interviews	6 h, 12 min	11 h, 48 min
Interviewees and their roles	1 Agile coach (A1) 1 Release Train Engineer (A2) 2 Scrum Masters (A3, A4) 2 Developers (A5, A6)	1 Manager (B1) 2 Release Train Engineers (B2, B3) 2 Product Owners (B4, B5) 2 Scrum Masters (B6, B7) 7 Developers (B8–B14)

Data were collected through observations consisted of photos and field notes. These observations were conducted by on-site visits every second or third month and lasted for two to five working days. During these visits, interviews were performed, and memoranda from meetings were studied. In total, 20 interviews were performed with key roles as well as team members who gave insights into multiple perspectives of the transformation (see Table 2).

The analysis followed a two-stage process of inductive and deductive coding of data, building upon the recommendations by Miles, Huberman, and Saldaña [20]. First, all field notes, meeting memoranda and interview transcripts were scrutinized and coded for the first time in an inductive manner. Initial codes were based on evidence of institutional logics at play. Secondly, a deductive coding of data was performed based on the six dimensions of Agile rulebook logic and Agile toolbox logic [15].

4 Findings

In this section, findings from Case A and Case B are presented. In each subsection, experiences of implementation strategies and perceived impacts of the transformation are displayed.

4.1 Case A

In Case A, five SAFe trained agile coaches helped in the transformation during the first one-and-a-half year. The Release Train Engineer [21] explained the view on implementing new methods: *"[The agency] always want a uniform way of working, a standardized way in the whole organization"* (A2).

Since SAFe was the baseline for all of them, the framework was, more or less, implemented by the book. One developer expressed: *"The agile coaches say 'According to SAFe…' or the reverse 'That's not something that SAFe says' as an excuse for not making decisions"* (A5). Another respondent (A3) claimed that the agile coaches were competing with each other regarding who knew the details of the framework best, rather than discussing possible tailoring or if some things could be discarded.

Financially, Case A used annual budgets and one planning event occurred just before the end of the year. Unfortunately, the annual budget process was delayed and, without a definite budget, managers were not allowed to plan for more than a couple of months into the new year. Still, the decided planning horizon of four sprints was kept, since SAFe describes a planning period of four sprints [21]. This meant that several teams were unable to plan for their final one or two sprints.

Several employees at Case A expressed a negative view towards the implemented roles and practices in the agile transformation. Many expressed that too much time was spent in meetings and that the agile way of working was not tailored to the work process. Employees also expressed limitations to team autonomy since they could not choose how to work as much as they had before the transformation. They also reported stress as a drawback, especially due to added work by following the detailed practices in SAFe forced on them by the agile coaches.

Despite the negative views, respondents expressed a better overview and transparency due to joint planning sessions with other teams as well as improved coordination and cooperation possibilities.

4.2 Case B

At Case B, one manager explicitly stated in an interview that they decided to implement roles, practices and routines suggested by SAFe based on their needs: *"Instead of implementing everything in SAFe, we decided to pick and choose practices that would help us"* (B1). Therefore, they did not ask consultants specializing in SAFe implementations for help. Instead, they implemented roles and practices on their own.

Practices were added and tailored along the way during the transformation process. PI planning was first implemented as suggested by SAFe, but was constantly tailored to supply the best planning overview to the teams: *"You get a much better understanding of the work ahead of you… all [PI] planning focuses on that, to visualize what you need to get done. You get a clear overview of what everyone is doing, and that is a huge advantage"* (B4).

A few of the employees expressed that too much time was spent in meetings and that the agile way of working was not entirely tailored to the work process, but not many compared to Case A. Instead, respondents expressed how the new way of working caused an increase in motivation and stress relief for the employees. Also, the new practices of joint planning sessions, PI planning and Scrum of Scrums [21], with several teams improved their planning precision.

Respondents at Case B also expressed a better overview and transparency, as well as improved coordination and cooperation possibilities. They also expressed that, although going through a transformation, there was no real interference on teamwork.

5 Discussion and Conclusion

Paasivaara et al. [22] presented the problems of large-scale agile transformations without the use of an agile framework. However, the study presented in this thesis shows that there are also risks when frameworks are implemented. In particular, there is a risk for suppressing tailoring if the framework becomes the norm by adhering to an *Agile rulebook logic*. This might be the case when there is too much agile coach support when all coaches are trained in a specific framework, as could be seen at Case A. According to respondents, too much of the discussions between coaches related to whether something was correct according to how it was described in SAFe. In Case A, commitment to method descriptions became the source of authority and method knowledge became the source of identity. Limitations to team autonomy and increased stress were reported impacts, which might be due to the *Agile rulebook logic* prevailing in Case A.

In Case B, roles, practices and routines were implemented and tailored continuously, piece by piece, along the way. Tailoring was based on the needs of the teams, such as PI planning where overview and transparency was in focus. Attentiveness and responsiveness became the source of identity and commitment to team decisions became the source of Authority. Increase in motivation and stress relief were reported impacts, which might be due to the prevailing *Agile toolbox logic* in Case B.

This study shows how two dichotomous institutional logics can be used for analytical purposes in large-scale agile transformation studies. The contribution of this paper is to show the possibilities of using two institutional logics for analyzing agile transformations.

There are, however, important limitations to this study. First of all, the study is conducted on a small dataset with only two cases investigated. Further studies on more organizations are necessary to confirm findings presented in this study. Especially, the observed cause-effect relationships between the identified logics and their impacts needs further confirmation. Also, it is important to remember that this is a case study of two organization based on a certain time, place, and the current individuals attending. Categorizing an institutional logic does not mean that a case organization is always anchored in that type [11]. Adherence to logics is fluid and changes over time. The changes may be due to changes in the world, or a result of a strategic decision [11].

References

1. Laanti, M., Kettunen, P.: SAFe adoptions in Finland: a survey research. In: Hoda, R. (ed.) XP 2019. LNBIP, vol. 364, pp. 81–87. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-30126-2_10
2. VersionOne. 14th annual "State of Agile Development" survey. http://www.versionone.com. Accessed 05 May 2021
3. Barroca, L., Dingsøyr, T., Mikalsen, M.: Agile transformation: a summary and research agenda from the first international workshop. In: Hoda, R. (ed.) XP 2019. LNBIP, vol. 364, pp. 3–9. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-30126-2_1
4. Moe, N.B., Holmström Olsson, H., Dingsøyr, T.: Trends in large-scale agile development: a summary of the 4th workshop at XP2016. In: XP 2016 Workshops: Scientific Workshop Proceedings of XP2016 (Article 1). ACM (2016)

5. Denning, S.: Strategic agility, using agile teams to explore opportunities for market-creating innovation. Strategy Leadersh. **45**(3), 3–9 (2017)
6. North, D.: Institutions, Institutional Change and Economic Performance. Cambridge University Press, Cambridge (1990)
7. Conboy, K., Carroll, N.: Implementing large-scale agile frameworks: challenges and recommendations. IEEE Softw. **36**(2), 44–50 (2019)
8. Šmite, D., Moe, N.B., Ågerfalk, P.J. (eds.): Agility across time and space: implementing agile methods in global software projects. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-12442-6
9. Thornton, P.H., Ocasio, W.: Institutional logics. In: Greenwood, R., Oliver, C., Sahlin, K., Suddaby, R.: The Sage Handbook of Organizational Institutionalism, pp. 99–128. SAGE Publications, Thousand Oaks (2008)
10. Friedland, R., Alford, R.R.: Bringing society back in: symbols, practices and institutional contradictions. In: Powell, W.W., DiMaggio, P. (eds.) The New Institutionalism in Organizational Analysis, pp. 232–263. University of Chicago Press, Chicago (1991)
11. Thornton, P.H., Ocasio, W., Lounsbury, M.: The Institutional Logics Perspective: A New Approach to Culture, Structure, and Process. Oxford University Press, Oxford (2012)
12. Rowlands, B.: Institutional aspects of systems development. In: Mills, A., Huff, S. (eds.) Proceedings of the 19th Australasian Conference on Information Systems, ACIS 2008, pp. 55–65. University of Canterbury (2008)
13. Doležel, M.: Possibilities of applying institutional theory in the study of hybrid software development concepts and practices. In: Kuhrmann, M., et al. (eds.) PROFES 2018. LNCS, vol. 11271, pp. 441–448. Springer, Cham (2018). https://doi.org/10.1007/978-3-030-03673-7_35
14. Berente, N., Hansen, S.W., Rosenkranz, C.: Rule formation and change in information systems development: how institutional logics shape ISD practices and processes. In: Bui, T.X., Sprague Jr., R.H. (eds.) Proceedings of the 48th Annual Hawaii International Conference on System Sciences, pp. 5104–5113. IEEE (2015)
15. Gustavsson, T.: Inter-team coordination in large-scale agile software development projects, Doctoral dissertation, Karlstad University, Karlstad (2020)
16. Schwaber, K., Beedle, M.: Agile Software Development with Scrum. Prentice Hall, Upper Saddle River (2002)
17. McBreen, P.: Questioning Extreme Programming. Addison-Wesley, Boston (2003)
18. Fitzgerald, B., Hartnett, G., Conboy, K.: Customising agile methods to software practices at Intel Shannon. Eur. J. Inf. Syst. **15**(2), 200–213 (2006)
19. Yin, R.K.: Case Study Research and Applications: Design and Methods, 6th edn. SAGE Publications, California (2017)
20. Miles, M.B., Huberman, A.M., Saldaña, J.: Qualitative Data Analysis: A Methods Sourcebook, 3rd edn. SAGE Publications, California (2014)
21. Scaled Agile Inc.: Scaled Agile Framework 5.0. http://www.scaledagileframework.org. Accessed 06 June 2021
22. Paasivaara, M., Behm, B., Lassenius, C., Hallikainen, M.: Large-scale agile transformation at Ericsson: a case study. Empir. Softw. Eng. **23**(5), 2550–2596 (2018). https://doi.org/10.1007/s10664-017-9555-8

9th International Workshop on Large-Scale Agile Development

Innovation in Large-Scale Agile - Benefits and Challenges of Hackathons When Hacking from Home

Rasmus Ulfsnes[1](✉), Viktoria Stray[1,2], Nils Brede Moe[1], and Darja Šmite[1,3]

[1] SINTEF, Trondheim, Norway
{rasmus.ulfsnes,nils.b.moe}@sintef.no
[2] Department of Informatics, University of Oslo, Oslo, Norway
stray@ifi.uio.no
[3] Blekinge Institute of Technology, Karlskrona, Sweden
darja.smite@bth.se

Abstract. Hackathons are events in which diverse teams work together to explore and develop solutions, software, or even ideas. Hackathons have been recognized not only as public events for hacking but also as a corporate mechanism for innovation. Hackathons are a way for established large-scale agile organizations to achieve increased employee wellbeing as well as being a curator for innovation and developing new products. The sudden transition to the work-from-home mode caused by the COVID-19 pandemic first put many corporate events requiring collocation, such as hackathons, temporarily on hold and then motivated companies to find ways to hold these events virtually. In this paper, we report our findings from investigating hackathons in the context of a large agile company by first exploring the general benefits and challenges of hackathons and then trying to understand how they were affected by the virtual setup. We conducted nine interviews, surveyed 23 employees, and analyzed a hackathon demo. We found that hackathons provide both individual and organizational benefits of innovation, personal interests, and acquiring new skills and competencies. However, several challenges such as added stress due to stopping the regular work, employees fearing not having enough contribution to deliver, and potential mismatch between individual and organizational goals were also found. With respect to the virtual setup, we found that virtual hackathons are not diminishing the innovation benefits. However, some negative effects surfaced on the social and networking side.

Keywords: Large-scale software development · Work-from-anywhere · Innovation · Hackathon

1 Introduction

Innovation in large companies is essential but is often harder to implement than in startups, where it is a natural and necessary part of the regular work. Large companies have many employees, many of them creative, but teams and team members are often bound to work only on the company strategy [1]. Creating an environment that fosters

P. Gregory and P. Kruchten (Eds.): XP 2021 Workshops, LNBIP 426, pp. 23–32, 2021.
https://doi.org/10.1007/978-3-030-88583-0_3

innovation and creativity requires employees to be motivated, and management needs to provide time and space for innovation to occur [1]. Therefore, to address the challenge of bringing new products to the market, many large software companies tend to approach innovation in a systematic way [1]. One example of such an approach is implementing an innovation program for internal startups [2]. For example, Google and Atlassian had their 20% time program [3], where developers are given 20% of their time to work on a project or initiative of their choosing. Another approach is to allocate days when all software developers in a company work on delivering a software product improvement of their choice. Innovation can be facilitated in a hackathon, a short and time-bound event, where participants work together to develop solutions, software or explore ideas. Large-Scale Agile organizations have increasingly utilized hackathons as a mechanism for innovation in the last 20 years [3–5].

Due to Covid19, company activities such as hackathons had been first postponed and eventually organized through virtual platforms such as MS Teams, Zoom, and Slack. Further, as many companies (Twitter, Spotify, Facebook, Salesforce) have announced their *Work from anywhere* (WFA) strategy, it is likely that structured innovative processes like hackathons will continue to be organized virtually in the future.

This unique situation has provided an excellent opportunity to understand how the shift from physical to a virtual hackathon affects the benefits and challenges of hackathons, but a andrge-scale agile organizations can embrace a WFA future. Even though there has been an increasing amount of literature on hackathons [7] and some literature on virtual hackathons [6], the literature regarding virtual hackathons for software companies, and especially the shift from physical to virtual, is scarce.

Motivated by the importance of innovation in large-scale agile and how large-scale agile organizations can embrace the WFA future, our research questions are:

RQ1: What are the benefits and challenges of hackathons?
RQ2: How are benefits and challenges affected by moving to a virtual hackathon?

Our chosen way to investigate this question was to perform a case study in a multinational software company utilizing large-scale agile methodologies. We observed the company employees during their virtual hackathon demo and inquired about the recognized benefits and challenges of hackathons for the company and their employees and how the virtual hackathon compared to the previous physical hackathons.

2 Background

Hackathons have been around for over 20 years, first appearing with OpenBSD and SUN microsystems in 1999 [8]. The word originates from the combination of the words “hack” and “marathon” [8]. There are multiple alternative names for hackathons, such as hackfest, jam, codefest, bug bash [5], as well as other more obscure names such as Delivery Day [4] and FedEx Day [3]. These events all contain similar elements, such as a limited and defined amount of time and the goal to create a minimum viable product (or at least something to show at a demo). Hackathons can be open events organized by universities, cities, municipalities, or internal corporate events.

Hackathons vary in how they are organized and executed, how ideas are structured, where it takes place (physical or virtual), and whether or not it is a competition [6]. These characteristics of hackathons provide different benefits and challenges. Falk Olesen and Halskov [7] emphasize that hackathon organizers need to tune its characteristics in order to achieve the wanted benefit for the target group.

Hackathons are not only a benefit for the organization, but also for the individual participants, providing an opportunity for individual development, goals and learning skills. There are also different categories of benefits identified in hackathons. Falk Olesen and Halskov [7] provide three categories*, structuring learning, structuring processes, and enabling participation.*

While hackathons provide benefits such as improving and fostering internal innovation, expanding competencies, and networking [3, 4, 6, 9–12], several challenges have been reported. Examples of challenges are prototypes that do not get sufficient follow-up work [7, 9, 11], stress by putting regular work aside [4] and issues associated with *hacker culture* and *low self-esteem* reported by Paganini and Gama [12].

Contradictions and tradeoffs are something apparent in the hackathon phenomenon, regardless of its formats. For example, diversification is both a benefit for learning and networking and a potential hinder for effective work during the hackathon [9]. Another contradiction is that the individual participants want to learn one specific technology or skill, disregarding whether or not that is useful for the organization [13].

3 Research Methodology and Approach

Our exploratory case study was conducted in a multinational agile software company identified here as “Ares” due to confidentiality. The company was founded in the early 2000s and develops content distribution software for mobile devices. Ares’s developers are located in two locations and are utilizing agile methodologies. The number of company employees at the time of conducting the study was 39 people. Ares organizes a voluntary hackathon every year and has done so for many years. Ideas are submitted and put up for voting based on managerial input. The employees then vote for three ideas they would like to work on. The hackathon teams are constructed based on the votes and managerial input to ensure suitable team sizes and diversity. The hackathon lasts two days, during which the teams self-organize, and at the end, all teams present a demo of their work. There are no awards or jury involved.

In this case study, we first analyzed a demo from a recent hackathon in the company. Based on the demo and previous research [3, 4], we developed an interview guide containing different areas for questions: *Background, Motivation, Idea generation, Cooperation, Hackathon organization, Expectations, Benefits, Challenges, and Virtual.*

Seven semi-structured interviews with people from four different development teams were conducted. They represented different skills and roles within the company, including developers, lead architects, senior engineers, and an advertisements operations manager, an overview of the informants can be seen in Table 1. The interviews were conducted in Norwegian or English, recorded, transcribed and qualitatively coded using descriptive coding, followed by a holistic overview and a thematic analysis [13].

The results were presented back to the company during an all-hands meeting to validate the qualitative findings and potentially elicit more information. In addition, during the meeting a survey was administered to the attendees asking them to rank the findings and add comments. Twenty-three persons responded to the survey.

Table 1. Informant overview

Role	Attended previous hackathons in Ares	Attended hackathons in other companies	Years at company
Android Developer	Yes	No	5
Data Scientist	Yes	Yes	3
Senior Developer	Yes	No	1
Android Developer	Yes	No	6
Software Architect	No	No	0.5
Advertisement Operations manager	Yes	No	2
Developer	No	No	1

4 Results

4.1 Hackathon Characteristics

In this chapter, we present our results by starting with a general description of hackathons, physical and virtual, which provide a context for understanding our findings.

How Often? – Once a Year. To our surprise, even though the participants were very positive towards hackathons, they did not want to conduct them too often. One participant explained, "It's *such a nice activity. It's something like your birthday. If it would be every month, it would become a bit more boring. It would be just another Hackathon."*

How Long? – Time Boxed. This is an inherent part of the hackathon, which often means that participants do not have time to finish the project. Opinions about time boxing differ. As one states, "*You should stop working when you are finished with the hackathon. Either you prove that something works, or you hopefully had fun."* This contrasts with those who said that the continuation of the hackathon idea was essential to them.

Which Ideas? – Whichever. The freedom to pitch and develop any idea gives a motivational boost and incentive to participate. As one explains, "*… with a more open approach, you can create ideas which might sound cool, but you can't really integrate it into your company. However, on the other side, those ideas might be something new which your company could work on."* Also, the opportunity to explore personal interests was important, one example being the idea of developing a guitar tuner popular among guitar players.

With Whom? – Whomever. The informants noted that the learning and teambuilding across disciplines and skills was a vital aspect of participating in the hackathon, indicating that a diverse team with different skill levels was good. However, this was contrasted by the importance for the participants to succeed. For example, one of the participants expressed, "*If I constructed the teams, I probably would have chosen different people.*"

What's at the End? – Demo, no Competition. One informant explained, "The *demo is everything about the hackathon. The hackathon is not successful if you do not have something to show at the demo.*" That being said, the focus on the demo can also make some people reluctant to join, as one stated: "*I haven't participated in hackathons earlier. I worried that I would not contribute enough because you need to deliver a product and present it at the end.*"

4.2 Benefits of Hackathons in Large-Scale Agile

Based on the interviews, we will now present the general benefits of hackathons categorized into individual and organizational. Finally, we report the benefits of most importance, based on the rating provided by the survey respondents (see Table 2).

Table 2. Rated benefits of hackathons

#	Individual - Benefits	Organizational - Benefits
1	Test new ideas fast	Teambuilding
2	Break from ordinary work	Ability to take advantage of emergent market opportunities
3	Fun from being with colleagues	Build a sense of belonging to the company
4	Build new stuff	Employees with broader skills and expertise
5	Build a network with colleagues	Increased marketability and talent attractiveness
6	Expand skills and expertise	Knowing who knows what (build employee network)

Our findings show that innovation is a key benefit both for the individuals and the organization, recognized as the ability to test new ideas fast and take advantage of emerging market opportunities. One informant said, "*It was a product idea I had wanted to develop for a while, and finally there was a hackathon so I could check if it worked.*" Other interviewees also mentioned that given the proper business case, hackathon products could be launched. One explained: "*Two years ago, we made a feature that was put directly into production, actually four weeks later, it was done properly. Hackathon proved it was possible, and it fits well into the existing product line.*"

Having a break from everyday work was highly valued by individuals. One interviewee explained: *"It's kind of relaxing for the people, for the company. Well, for the team itself because you are not doing exactly what you are paid for."*

Teambuilding was rated as the top organizational benefit. Having fun from being with colleagues was rated third as the individual benefit. One informant explained, *"In hackathons, what normally happens is that the main outcome, at least in our experience, tends to be the social aspect of it, the teambuilding."* Another stated, *"you will remember other important parts of the hackathon than just the work or who was the winner, for example, the fun you had with your colleagues and the knowledge you gained."*

Increased marketability and talent attractiveness are achieved due to the individual benefits and because the hackathon in itself is a company practice that gives an edge during the recruitment process. The individual expands their skills and expertise by working on topics and technology where the participant did not have much prior knowledge and thus utilizing the hackathon for learning, providing the organization with higher and broader skilled employees. Building network among colleagues and having employees that know who knows what, increasing the inter-team communication, and give employees new insights into and respect for one another's role and work.

4.3 Challenges of Hackathons in Large-Scale Agile

Even though there are clear benefits of hackathons, employees also mentioned some challenges. Therefore, we mention the challenges corroborated by the survey. The complete list of challenges is shown in Table 3.

The top individual challenge was related to hackathons adds stress. Even though hackathons are mostly a welcomed break, the timing for running a hackathon is not always good. As someone explained, *"you have pressure during your workday, and you need to get something finished and you look at the hackathon as something in the way that is just disturbing you from important work."* The two organizational challenges echo this challenge – stopped production for a time and increased employee stress.

Mismatch in skills and desires due to the team diversification was another top challenge. The optimal team structure was not always obvious. One participant described that the team members did not always have the same desires and motivation for participating in the hackathon. One interviewee explained, "*It was challenging that we had different goals we wanted to achieve. I wanted my product to be launched while the others wanted to expand their skills on machine learning.*"

Being unable to complete the hackathon projects is somewhat frustrating. Even though the employees are well aware that the hackathon is timeboxed, they still desire to finish their products. A participant explained, "*I am not allowed to work on it, I have other work tasks. That's a downside with hackathons."* This results in developers feeling dissatisfied as their goals remain unfulfilled. As someone explained, "*You know, the worst feeling you can have is when you write code and after some time it gets forgotten and no one ever uses it."*

4.4 Virtual Hackathons – What Are the Changes?

A virtual hackathon does not impact the ability to take advantage of emerging market opportunities achieved through hackathons. The idea generation and idea voting are similar to the previous hackathons, the difference is that the teams work and collaborate virtually. As one says, *"But from a technical point of view, to just develop something, you* *can always use Slack or whatever, or just Google meet and call."* In addition to this, our findings show that the number of hours spent on virtual hackathon is less than when it is held physically. "*During the first hackathon we stayed in the office up until midnight.*" In contrast, in the virtual hackathon, the employees started by dividing the work, then worked focused individually, continuously coordinating through Slack and using virtual meeting rooms for discussions and planning: *"We did the first meeting when we split the work. So, after a few hours we met again. We have this approach, let's just give it a try. Then a few hours later, we had another meeting, and we tracked our progress quite often."*

A break from everyday work through hackathons is especially welcome after stressful periods such as migrations, or as one says in a pandemic," Especially *in these pandemic times. You know we are working from home. So, we want to meet people in real life and hackathons are one way to just, you know, meeting and collaborating".*

We see a reduction in the benefit of having fun with colleagues: "*It was not as fun as it used to, hackathons are usually more fun than over video during Covid, in my own home.*" In addition, most employees mentioned that they would prefer to have the hackathon co-located in the same physical location in order to socialize and have a good time. Lastly, disagreements due to mismatches in individual goals seem to be harder to solve through virtual meetings.

Informants note that there is less tendency to work after hours during a virtual hackathon. This was explained due to people logging off after their part of the project was done, *"Some years, when we had pizza and social interactions, I worked to midnight. This year, I worked to 9 pm, other team members (designers) only worked to 4 pm, and it did not make sense for them to hang around on Slack after they were finished. In the office it's different because you can be social".*

5 Discussion

Noting the lacking research on internal corporate hackathons [9], we will now discuss our research questions. Our first research question was, "What are the benefits and challenges of hackathons?" Consonant with [3] and [4], we found that providing dedicated days for developers may lead to innovation and new product development and that there are benefits both on the individual and company levels, as seen in Table 1. Developer involvement, productivity, competence, and wellbeing are thus important factors for large-scale agile companies that foster innovation. We also found that learning and networking are important benefits, as corroborated by [7, 12]. In addition, just providing time to take a break from regular work and having fun is important for the wellbeing of the employees. In addition to the different levels of benefits, we also see a clear link between the characteristics of hackathons and the benefits that emerge from them. We also find tradeoffs between some of the benefits. For example, building new stuff vs. teambuilding as our findings show that tailoring the teams for diversification might take away the team's ability to perform during the hackathon, corroborating Falk Olesen and Halskov [7] where the skills wanted by the individual did not match that of the organization. Our findings showed that some employees were frustrated when their product was not continued. This confirms a challenge with hackathon continuation, as reported in previous research [5, 11]. We also found additional challenges such as taking time away from everyday work, whether it was good timing or not, corroborated by [4]. Personal desires and goals could also pose a challenge where teams could not agree on common goals. Fear of not contributing to the team during the hackathon was also found due to low self-esteem, this corroborates Paganini and Gamas [12] findings, albeit our findings are provided by male developers with limited experience, showing that low self-esteem is not only found with female participants but also in-experienced participants.

Our second research question was, “How are benefits and challenges affected by moving to a virtual hackathon?” Our results suggest that the shift from physical to virtual hackathons in large-scale agile does not seem to change the innovation benefits or worsen the challenges that much. We found that the number of hours spent on the hackathon decreased compared to physical hackathons, and the work was more divided and focused between the participants. The employees were producing good quality demo prototypes and were happy with the outcome. The findings suggest that having fun with colleagues has decreased in virtual hackathons since physical hackathons provide more encouragement and room for socializing, which was not provided in the virtual hackathon.

5.1 Limitations

This study only has one sample for physical to virtual hackathons. In addition, all the interviewees were male, and we did not elicit information about female participants. Qualitative studies are also subjective by nature, however, the survey we administered provided some quality assurance of the validity of the findings.

6 Conclusion and Future Work

Our results suggest that virtual hackathons in large-scale agile organizations benefit individuals’ opportunity to test new ideas, take a break from regular work and acquire new skills as well as broader skillset. In addition, hackathons provide the organization with new products, refreshed employees, and a more competent workforce. Virtual hackathon teams develop good technical solutions; however, the social and fun aspects are diminished compared to physical hackathons.

Practitioners should also be aware that challenges can include additional stress and dissonance in hackathon teams due to differences in experience and personal goals.

Future research should focus on collecting more data about virtual hackathons in other large-scale agile organizations to develop further insights into how virtual hackathons can be organized and improved. When doing this, one could consider using the SPACE framework for developer productivity suggested by Forsgren et al. [14]. The framework recognizes multiple productivity dimensions for large-scale agile organizations, both on an individual, team, and organizational level, that are similar to our findings on hackathons.

References

1. Edison, H., Wang, X., Jabangwe, R., Abrahamsson, P.: Innovation initiatives in large software companies: a systematic mapping study. Inf. Softw. Technol. **95**, 1–14 (2018)
2. Sporsem, T., Tkalich, A., Moe, N.B., Mikalsen, M.: Understanding Barriers to Internal Startups in Large Organizations: Evidence from a Globally Distributed Company. ArXiv Prepr. ArXiv210309707 (2021)
3. Moe, N.B., et al.: Fostering and sustaining innovation in a fast growing agile company. In: Dieste, O., Jedlitschka, A., Juristo, N. (eds.) PROFES 2012. LNCS, vol. 7343, pp. 160–174. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-31063-8_13
4. Barney, H.T., Moe, N.B., Dybå, T., Aurum, A., Winata, M.: Balancing individual and collaborative work in agile teams, pp. 53–62 (2009)
5. Komssi, M., Pichlis, D., Raatikainen, M., Kindstrom, K., Jarvinen, J.: What are Hackathons for? IEEE Softw. **32**(5), 60–67 (2015). https://doi.org/10.1109/ms.2014.78
6. Kollwitz, C., Dinter, B.: What the hack?–towards a taxonomy of Hackathons, pp. 354–369 (2019)
7. Falk Olesen, J., Halskov, K.: 10 Years of Research With and On Hackathons (2020). https://doi.org/10.1145/3357236.3395543
8. Briscoe, G., Mulligan, C.: Digital Innovation: The Hackathon Phenomenon, p. 13
9. Nolte, A., Pe-Than, E.P.P., Filippova, A., Bird, C., Scallen, S., Herbsleb, J.D.: You hacked and now what? -exploring outcomes of a corporate Hackathon. In: Proceedings of the ACM Human-Computer Interaction, vol. 2, no. CSCW, pp. 1–23 (2018)
10. Flores, M., et al.: How can Hackathons accelerate corporate innovation? In: Moon, I., Lee, G.M., Park, J., Kiritsis, D., von Cieminski, G. (eds.) APMS 2018. IAICT, vol. 535, pp. 167–175. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-99704-9_21
11. Nolte, A., Chounta, I.-A., Herbsleb, J. D.: What happens to all these Hackathon projects?. In: Proceedings of the ACM Human-Computer Interaction, vol. 4, no. CSCW2, pp. 1–26 (2020). https://doi.org/10.1145/3415216
12. Paganini, L., Gama, K.: Engaging women's participation in Hackathons: a qualitative study with participants of a female-focused Hackathon. In: International Conference on Game Jams, Hackathons and Game Creation Events 2020, Osaka Japan, pp. 8–15, August 2020. https://doi.org/10.1145/3409456.3409458
13. Saldaña, J.: The Coding Manual for Qualitative Researchers, 2nd edn. SAGE, Los Angeles (2013)
14. Forsgren, N., Storey, M.-A., Maddila, C., Zimmermann, T., Houck, B., Butler, J.: The SPACE of developer productivity. Queue **19**(1), 20–48 (2021). https://doi.org/10.1145/3454122.3454124

Impacts of COVID-19 Pandemic for Software Development in Nordic Companies – Agility Helps to Respond

Petri Kettunen[1](✉), Tomas Gustavsson[2], Maarit Laanti[3], Andreas Tjernsten[4], Tommi Mikkonen[1], and Tomi Männistö[1]

[1] University of Helsinki, Helsinki, Finland
{petri.kettunen,tommi.mikkonen,tomi.mannisto}@helsinki.fi
[2] Karlstad University, Karlstad, Sweden
tomas.gustavsson@kau.se
[3] Nitor Delta, Helsinki, Finland
maarit.laanti@nitor.com
[4] Nitor Agile AB, Stockholm, Sweden
andreas.tjernsten@nitor.com

Abstract. In 2020, the global COVID-19 pandemic affected almost every company in some way also in the Nordic countries. Depending on the different industry sectors of the companies, the impacts have varied from minor risks to severe disruptions but also even booming businesses. In all, agility and resilience have been required to continue and even to survive. In 2018, we started conducting large-scale agile surveys in Finland and Sweden. For the 2020 survey round, we included questions about the current pandemic situation impacts and how agility has helped to respond. The respondents represented software professionals from different industries, not limited to information and communication technology (ICT) companies. The results indicate that although the perceived impacts have mostly been negative (53%), it is not all so. One-third (33%) reported positive impacts such as increased business and better well-being. The majority (55%) of the responses indicated that agility has helped to respond to the pandemic situation. Remarkably, 59% reported that their companies have improved agility during the past year. Improved agility appears to be positively related to the ability to respond to the pandemic. We did not discover significant differences between the Finnish and Swedish respondent cohorts.

Keywords: Agility · COVID-19 · Digitalization · Agile · Scaled agile

1 Introduction

In 2020, the COVID-19 disease spread across the world and was declared a "Public Health Emergency of International Concern" by the World Health Organization (WHO). Governments enforced regulations and proposed recommendations to prevent further spread of the virus. Technology companies all over the world locked down their offices and made their employees work from home. Many businesses were suddenly disrupted.

P. Gregory and P. Kruchten (Eds.): XP 2021 Workshops, LNBIP 426, pp. 33–41, 2021.
https://doi.org/10.1007/978-3-030-88583-0_4

We investigated the pandemic phenomenon based on a recent survey. The survey was conducted during October–November 2020. The overall purpose of the survey was to understand the state of agility in the Nordic countries. This study aimed at answering how the current situation of the pandemic has impacted companies and how well agility helps companies to respond to the situation.

2 Background

Current studies reflect impacts on software professionals during the first months of the COVID-19 pandemic. Working from home at a scale never seen before under these new conditions is very much different from normal working from home. In one survey among software professionals, the results show evidence of declining productivity and well-being [1]. They also found an indication of a disproportionately negative effect on women, parents, and people with disabilities. Another investigation found that working from home during the pandemic has different impacts on the productivity of software professionals, depending on which metrics to use [2]. The impacts also differed based on project type, size, and age of employees. A study of GitHub activities presents that patterns of activity of software professionals might have implications for burnout [3]. In addition, the study suggests that the cadence of work has changed since working days have become longer by up to an hour a day, on both weekdays and weekends.

Overall, the pandemic has affected different industries and companies in different countries in various ways, even within the core ICT industry, and different companies have devised various ways to respond and adapt to the impacts (e.g., [4, 5]). The pandemic year 2020 affected significantly and suddenly the work-life in Finland and Sweden too, due to extensive telecommuting recommendations. Full-time remote working has not been typical in either Finnish or Swedish companies in general.

Because agile companies and software organizations are by definition capable of coping with and adapting to changes in their environments and operations, we expected that they were able to respond to the impacts of the pandemic in fitting ways [6, 7]. Moreover, some companies may even have been able to deal with the disruptions as new opportunities causing positive impacts. However, agile practices are based on dedicated teams that interact closely with each other and the customer. Teams collaborating in close proximity and communicating in daily face-to-face meetings are typical characteristic of agile. But since remote working does not allow agile practices to be performed in the previous way, practices and tools needed to be replaced. This significantly changed the agile way of working [5].

3 Research Design and Method

The research effort started in Finland in 2018 from an industrial stance. The initial idea of this survey research endeavor was to investigate the current state of agile software development in Finnish companies. We were interested in refreshing how companies currently use agile methods and how agile they really are. In 2018 and 2019, we conducted two survey rounds, the first year in Finland and the second in Sweden. We have reported those results in our prior publications [8]. The results in this paper do not include answers from those earlier rounds.

Overall, the purpose of our survey research was to examine the current state of agile development and enterprise agility. Different companies may approach agile development and agility in different ways. Hence, we were interested in examining how agile companies really are nowadays and how they currently practice agile software development. We are interested in measuring how widely agile methods and practices are currently applied in industrial practice and how that is evolving. Moreover, we want to go beyond team levels to large-scale agile and enterprise agility. We seek to understand why different companies want to change – even transform – with agile means and how beneficial and successful their particular changes have been. In all, we targeted to investigate not just ICT companies but industries in general. The target population was intentionally not limited to software companies since we were also interested in non-software companies (i.e., companies in other industries than IT) currently facing digitalization and becoming more software-intensive. We were also interested in the future. We aimed to investigate not only the current whereabouts but also the future intentions of the companies.

The research method was a descriptive survey. The questions and the predefined answer choices were compiled by referring to selected prior surveys and by deriving from own experiences and research. Most of the questions were closed-type with an open free-text choice. Certain questions depended on their preceding questions. The draft questionnaire was first piloted both in our industrial and academic organizations.

In 2020, we revised the questionnaire based on the experiences of the 2018–2019 survey. We added some new questions, removed some of the previous ones, and refined some. In particular, we introduced a section of the current situation of the pandemic, which emerged during the questionnaire revision period. The revised version of the questionnaire comprised in total 48 question items with Finnish, Swedish, and English language variants (translated by the native Finnish and Swedish speaking authors of this paper). Again, certain of them depended on the selector question answers. The content questions (except the selector ones) were non-mandatory and had “I don’t know” and N/A options. A pilot round was conducted.

For data collection, the survey was implemented with a web-based online questionnaire tool. We considered several potential distribution channels in order to reach a wide, representative sample population. However, due to pragmatic constraints, we decided to use convenience sampling. The questionnaire was distributed through a Finnish consulting company newsletter to people (300) mainly in Finland and Sweden who are interested in the company’s offering of software consultancy, training, and agile transformation services, as well as shared through several social media channels, especially by posting links to the survey in agile user-groups on LinkedIn and Facebook. In addition, we advocated it to certain software research interest groups. The survey was open for seven weeks during October and November 2020. We received 137 responses.

4 Results

To address the research objectives described above, we analyzed the respondents’ answers to the following survey questions:

- Q0: *How do you see that Your company's overall agility has changed during the past year?*
- Q1: *How much and in what way does the current situation of global pandemic impact Your company?*
- Q2: *How has the current situation of global pandemic impacted Your company?*
- Q3: *How well does agility help Your company to respond to the situation?*

Figure 1 illustrates the basic descriptive statistics of the respondents and organizations. 54% of the respondents were located in Finland, 43% in Sweden. Software development and software process development were the most frequently reported roles. 66% of the respondents are in large or very large companies. Notably, two thirds (67%) of the respondents are in other sectors than core ICT businesses (computer programming, consultancy, and related activities). Agile methods are widely used.

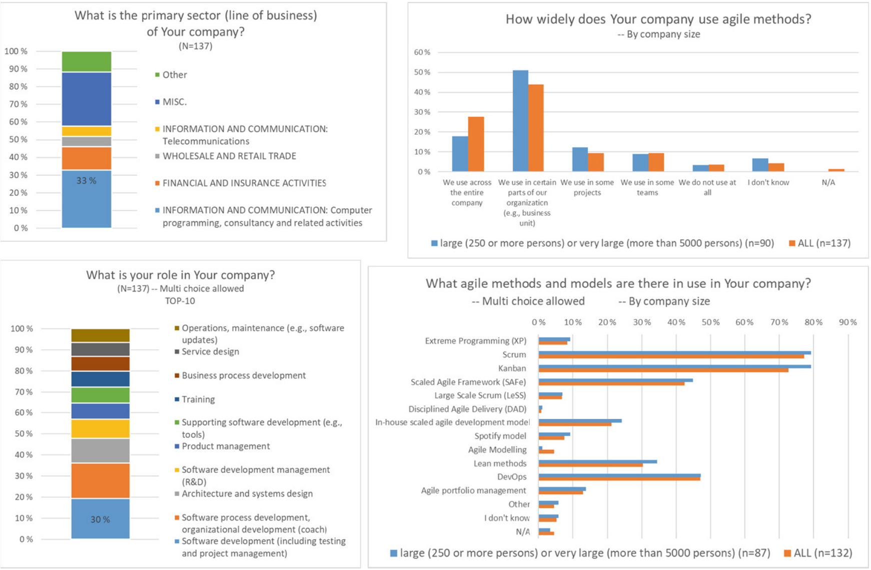


Fig. 1. Demographical information of the respondents and their organizations.

4.1 How Companies' Overall Agility has Changed (Q0)

The results about how the respondents perceived that the company's overall agility has changed during the past year (i.e., at the time of the survey in October–November 2020) reported were as follows (n = 132 (87 of large/very large companies)): 59% (69%) improved (42% (46%) a little, 17% (23%) significantly), 8% (6%) declined (a little or significantly), and 23% (20%) remained the same. Notably, the majority reported improved agility. The distribution in large/very large companies was similar to the all. There appeared to be no significant differences between Finland and Sweden.

4.2 How Much and in What Way the Current Global Pandemic Impacts (Q1)

The purpose of this question is to probe, how positively/negatively the pandemic was perceived to impact (answer scale [extremely neg., extremely pos.]). Aggregating the results, about a half (53%) of the respondents (n = 135 (89 of large/very large companies)) reported negative impacts while one third (33% (34%)) reported positive impacts (3% (2%) even extremely). The distribution in large/very large companies was similar to the all. In all the distribution was slightly more on the positive side in Finland than in Sweden (Finland 51% negatively, 36% positively; Sweden 58% negatively, 29% positively).

The ICT sector reported being impacted more negatively (50% negatively, 39% positively) than the other main sectors of the respondents. Interestingly, the wholesale and retail trade sector appear to have experienced more positive impacts (38% negatively, 63% positively), while the financial and insurance sector reported equally positive and negative impacts (39% negatively, 39% positively).

4.3 How the Current Situation of Global Pandemic has Impacted (Q2)

Following the Q1 (Sect. 4.2), Table 1 displays how the pandemic situation affected companies in practice. The responses to this question are open comments answered by 80 of the respondents. The remaining respondents (Q1 n = 135) did not answer this question. The replies (Finnish and Swedish translations into English) were coded in twelve descriptive themes, and the themes were further grouped as positive (POS), negative (NEG), and miscellaneous (MISC) ones, based on the impact on the company. The percentages displayed in Table 1 are based on the number of answers coded for each theme and should be seen as an indication of differences in reported impacts.

Notably, there are both positive and negative impacts reported on business and well-being. A declining business was the most often reported impact both in Finland and in Sweden, emphasized in Finland.

An increase in remote working was also reported noticeably more often in Finland. One might expect answers relating to remote work (*Remote working increased*), and this survey showed this theme with the second highest number of answers. The reported negative impact of *Cooperation difficulties* was often perceived as an effect of being forced to work from home, while a positive impact was that the situation has in several organizations led to *Improved digitalization.*

Several answers categorized in the theme *Declining business* show different amounts of lost customer projects and sales. Some answers show that one impact is an *Impeded development* situation, implying that improvement activities have been halted. Also, *Layoffs and reduction* of work hours were reported. However, the survey shows several positive changes for some organizations. These include an *Increased business* situation while some organizations perceived *No big change*, that business could continue as usual.

Some responses showed a notion of *Declining well-being* regarding problems of lowered job satisfaction and well-being. It is clear that the experience differs, however, since some respondents perceived *Increased well-being.*

Table 1. (Q2) How has the current situation of global pandemic impacted Your company?

Themes	Type	% of responses		
		ALL (n = 80)	Finland (n = 39)	Sweden (n = 41)
Declining business	NEG	18%	23%	12%
Cooperation difficulties	NEG	8%	5%	10%
Impeded development	NEG	8%	5%	10%
Layoffs and reduction	NEG	6%	5%	7%
Declining well-being	NEG	6%	3%	10%
Increased business	POS	8%	8%	7%
Improved digitalization	POS	5%	3%	7%
Increased well-being	POS	4%	3%	5%
New insights	POS	3%	3%	2%
Remote working increased	MISC	15%	23%	7%
New work practices	MISC	15%	18%	12%
No big change	MISC	6%	3%	10%

The theme *New work practices* show how organizations have been forced to find new solutions based on the pandemic. The answer "We had to plan new business concepts" shows signs of profoundly affected organizations. Answers also showed a perceived impact of *New insights* in the organization. One answer, for example, expressed a "Demand for truly agile consultancy raising".

4.4 How Well Agility Helps to Respond to the Situation (Q3)

Considering the perceptions of how well agility helps to respond to the pandemic situation (scale [not much, very well]), the majority (55% (56%)) of the responses (n = 134 (89 of large/very large companies)) was on the supportive side (from the scale midpoint to "very well"), while 12% (13%) was from the scale midpoint to "not much". The distribution in large/very large companies was similar to the all. Notably, 16% (17%) reported that they do not know (11% of the Finnish respondents (n = 71) and 24% of the Swedish respondents (n = 59)).

4.5 Further Insights

Having presented the direct results of the survey, in this subsection, we take a deeper look at the different questions Q0–Q3 and analyze them in combination. This also informs further research (Sect. 5.4).

In order to investigate to which degree agility has helped organizations based on how challenging their impact of the pandemic was, we cross-analyzed the two questions Q1 and Q3, i.e., the relationship between the direction the pandemic impacted companies and how well agility has helped them to respond to the situation. It showed most respondents perceiving agility to have been helpful regardless of how (NEG or POS) the pandemic had an impact. To verify if there is a significant difference, we used the Kruskal-Wallis H test which is a non-parametric alternative to One Way ANOVA. However, the Kruskal–Wallis H test does not show a statistically significant difference (H = 13.112, df = 10, Asymp. Sig. = 0.217). This means that the agile way of working has been considered as helpful both in organizations where the impact of the pandemic has been negative as well as positive.

By cross-analyzing the two questions Q0 and Q3, i.e., how companies' overall agility has changed during the past year and how well agility has helped them to respond to the situation, we also investigate a possible relation. A majority of respondents perceived agility to be helpful during the pandemic, and a majority have improved their companies' overall agility. The Kruskal–Wallis H test confirmed a statistically significant difference (H = 15.278, df = 4, Asymp. Sig. = 0.004), which means that respondents perceiving agility to be helpful in this situation have also improved their agility during the last year.

To summarize the perceived impacts, although the respondents have expressed both positive and negative impacts, a majority of the answers show negative perceived impacts to the organization.

5 Discussion and Conclusions

We assess our study, derive recommendations, and conclude with further work plans.

5.1 Related Works

In comparison to our survey question Q3 (Sect. 4.4), a recent industrial study reported that mature agile business units have been more successful (in terms of customer satisfaction, employee engagement, operational performance) in coping with the impacts of the pandemic crisis than non-agile ones [7]. Following established agile practices and cross-functional ways of working, they were able to quickly reprioritize the business targets and continue. We found more negative impacts (53%) due to the pandemic reported from our respondents than positive (33%).

The majority of our respondents were in large or very large companies. In contrast, in a recent study, one software startup company faced many uncertainties and demands for adaptation when it was forced to quickly change the previous co-located agile ways of working to remote working at home [4]. The key challenges were maintaining the teams' productivity and ability to continue delivering satisfying customer value. Furthermore, the company paid attention to employee well-being. Well-being was also one of our resulting themes in Q2 (Sect. 4.3).

An immediate impact of the pandemic for almost every company has been increased remote work – even changing to full-time work at home due to governmental rulings. [1]. An increase in remote working was also one of our observations (Sect. 4.3).

Considering agility development (Q0, Sect. 4.1), a German management survey discovered that perceived agility of projects during the early stages of the pandemic remained high and, on average, slightly increased [5]. This concurs with our findings.

5.2 Recommendations

Our study suggests that developing agility is useful both for and during turbulent times. It is Important to discern the type of change (internal or external) and its dynamic nature (factors increasing or decreasing, possibly even in both ways).

Business Agility is, by definition, an ability to sense and respond to changing business conditions quickly. Agility is also a mindset to react and move faster. This may be an explanation why the organizations that are agile and that have agile mindset have been coping better with the changing conditions due to the pandemic: the organization's structures and decision-making already support reacting quickly to changing conditions. The personnel is rather rewarded for taking action than punished for initiatives.

Furthermore, innovation is an inherent element of agile business and, under the pandemic circumstances, its role may have been amplified in companies to continue their businesses. However, such factors as extensive remote working and consequent co-operation difficulties discovered in this survey data (Sect. 4.3) may have had negative impacts on innovation performance.

5.3 Limitations and Threats to Validity

We did not ask directly about large-scale agile. However, 66% of our respondents were in large/very large companies and scaled agile methods are often used (Fig. 1).

We have recognized the limitations and potential threats earlier [8]. Especially, a limitation is that we did not ask the respondents to identify their organizations. Therefore, we cannot tell the number of different organizations in our respondent population, and we refrain from judging how representative our industrial sample is. Due to such statistical validity limitations, we make no attempt at generalizing the findings.

A construct validity concern is whether all the respondents in different roles and companies have interpreted all the terms in our questionnaire in the same way (e.g., 'agility'). We do not consider internal validity to be a significant concern since the purpose of the survey is primarily exploratory rather than explanatory. We have thus been cautious not draw decisive conclusions in this study. External validity judgement is limited by the background information collected. Research comparisons with other industrial surveys should consider possible biases. Due to the company-specific call-out of the survey (see Sect. 3), sampling bias is a threat. With the social media distribution, the response rate is unspecified. Considering reliability, the main concerns in this survey are thus the formulation of the question items and the dissemination.

5.4 Further Research

The mindset and culture of the agile organization is a possible area for further research based on our survey data. Related to that, one of our survey questions was: *To what extent is Your company culture supporting agile ways of working, methods and practices?* This question could give insights into whether a supporting company culture for agility has helped in managing the pandemic.

Another area for future research, based on our survey data, is to investigate answers to the survey question: *Is the company as agile as it should be?* Cross-analyzing this question with the perceived impacts (Sect. 4.2) could bring further understanding to whether the agile maturity has had an impact on managing the pandemic (Sect. 4.4).

Truly agile companies may have been more successful in continuous innovation. In this questionnaire we had the following multi-choice question related to that: *What goals does the company attempt to achieve by agile means?* One predefined answer choice was: *New business (product and service innovation).* There may be new business opportunities following the recovery of the pandemic to a "new normal".

References

1. Ralph, P., et al.: Pandemic programming: how COVID-19 affects software developers and how their organizations can help. Empir. Softw. Eng. **25**(6), 4927–4961 (2020)
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 COVID-19 Pandemic. arXiv preprint arXiv:2005.13167v2 (2020)
3. Forsgren, N.: Octoverse spotlight: an analysis of developer productivity, work cadence, and collaboration in the early days of covid-19. https://github.blog/2020-05-06-octoverse-spotlight-an-analysis-of-developer-productivity-work-cadence-and-collaboration-in-the-early-days-of-covid-19/. Accessed 9 March 2021
4. da Camara, R., Marinho, M., Sampaio, S., Cadete, S.: How do Agile Software Startups deal with uncertainties by Covid-19 pandemic? Int. J. Softw. Eng. Applications (IJSEA) **11**(4), 15–34 (2020)
5. Schmidtner, M., Doering, C., Timinger, H.: Agile working during COVID-19 pandemic. IEEE Eng. Manag. Rev. https://doi.org/10.1109/EMR.2021.3069940
6. Worley, C.G., Jules, C.: COVID-19's uncomfortable revelations about agile and sustainable organizations in a VUCA world. J. Appl. Behav. Sci. **56**(3), 279–283 (2020)
7. Handscomb, C., et al.: An operating model for the next normal: lessons from agile organizations in the crisis. https://www.mckinsey.com/business-functions/organization/our-insights/an-operating-model-for-the-next-normal-lessons-from-agile-organizations-in-the-crisis. Accessed 13 Feb 2021
8. Kettunen, P., Laanti, M., Fagerholm, F., Mikkonen, T., Männistö, T.: Industrial agile transformations lacking business emphasis: results from a Nordic survey study. In: Klotins, E., Wnuk, K. (eds.) ICSOB 2020. LNBIP, vol. 407, pp. 46–54. Springer, Cham (2021). https://doi.org/10.1007/978-3-030-67292-8_4

The EFIS Framework for Leveraging Agile Organizations Within Large Enterprises

Alexander Poth[1](✉), Mario Kottke[1], Christian Heimann[1], and Andreas Riel[2]

[1] Volkswagen AG, Berliner Ring 2, 38436 Wolfsburg, Germany
{alexander.poth,mario.kottke,christian.heimann}@volkswagen.de
[2] Grenoble Alps University, Grenoble INP, G-SCOP, CNRS, 38031 Grenoble, France
andreas.riel@grenoble-inp.fr

Abstract. This article presents the design and application of the EFIS framework that combines four pillars to foster agile and lean working in organizations within large enterprises. These pillars constitute the empowerment of teams, the focus on products, the integration of processes, and the scaling of knowledge. The framework is designed to systematically address typical large enterprise challenges such as governance of regulation requirements and product risks. By design, EFIS is lean and nimble to make it easily adaptable to domain-specific demands within large organizations. It can be used as a stand-alone approach to establish and continuously improve lean and agile organizations, as well as in combination with existing approaches like SAFe®.

Keywords: Large-scaling agile · Agile transformation · Agile framework

1 Motivation, Context and Methodology

In the ongoing trend of the agile transformation of large companies, the set of established large-scale lean and agile frameworks has grown to a considerable number. The most established ones in practice are Scaled Agile Framework (SAFe®) [1], Large-Scale Scrum LeSS [2], Nexus [3] and Spotify [4]. Domain-specific challenges have led to specialized methodologies such as R-Scrum [5] and SafeScrum® [6] for safety. Additionally, agile organizations should address autonomy, mastery and purpose adequately [7].

Analyzing these frameworks, we identified the following shortcomings:

(1) The governance in terms of a reliable chain of accountability, responsibility, shared-responsibility, mastery and autonomy is only vaguely addressed.
(2) Measuring the progress of the adoption of framework practices by the teams and the overall organization is not clearly covered by indicators and methods.
(3) Scaling of the framework practices without significant coaching and training efforts is not explicitly addressed.

P. Gregory and P. Kruchten (Eds.): XP 2021 Workshops, LNBIP 426, pp. 42–51, 2021.
https://doi.org/10.1007/978-3-030-88583-0_5

This paper proposes the EFIS (Empower, Focus, Integrate & Scale) framework that aims at addressing these shortcomings from a holistic perspective. It has been designed with a bottom-up design science research approach [8] within the context of the Volkswagen Group IT. Over a timespan of five years, several building blocks have been designed, implemented, evaluated and improved. Their proven-in-use designs have been integrated in a larger framework, paying close attention to the consistency between blocks in order to ensure a holistic perspective when addressing the organizational challenges. The validation of the entire framework has been done implicitly through the validation of each building block in at least two different organizational units, as well as by measuring several teams' agile maturity progress over time.

Section 2 presents the literature and established agile frameworks. Section 3 elaborates on the architecture and application of EFIS, as well as its key characteristics. Section 4 explains how EFIS helps implementing accountability at large-scale through mastery within the scope of an agile team. Section 5 reports on how EFIS has been implemented at the Volkswagen AG, and critically evaluates success. Section 6 discusses the limitations. Finally, Sect. 7 concludes by summarizing the article's key contributions to research and practice and giving an outlook to the authors' ongoing research activities.

2 Established Agile Frameworks and an Literature Overview

One of the SAFe® [9] core values is "Built-In Quality", however, the focus is on product quality that is achieved through testing and/or design for quality. The quality management aspect of continuous improvement is part of the learning culture with a modified PDCA cycle [10]: the A stands for *adjust* instead of *act* in the original plan-do-check-act cycle. A core concept of LeSS [2] is to reduce organizational complexity. As part of technical excellence, LeSS focuses on testing in terms of test-driven development, thinking about testing, unit testing, as well as acceptance testing. LeSS explicitly eliminates support groups like "quality and process" as potential bottlenecks [11]. Nexus [3] is based on Scrum and defines additional accountabilities to the roles. However, it does not explicitly address governance, compliance and quality. Nexus can be seen as the enhancement of enterprise Scrum (eScrum). Scrum@Scale™ [12]: It is an agile scaling framework based on Scrum [13] and scales with the Scrum of Scrum (SoS) approach. It does not explicitly address aspects of governance and quality either. The Spotify Model is not a scaling framework by design, but rather an agile organizational building block kit [4]. Accountability is realized by the product life-cycle and features end-to-end responsibility. Furthermore, the concept of alignment enabling autonomy is used as a base for different Squads to work cooperatively on features, infrastructures or client applications. Recipes for Agile Governance in the Enterprise (RAGE) [14] are an approach focusing on making decisions repeatable and transparent. It distinguishes project, program and portfolio level. It uses Scrum and Kanban as a base for the governance extensions called recipes. Recipes are built on roles, ceremonies, artifacts, metrics and governance points. They can be combined with SAFe® on the program level. For implementation RAGE offers a white paper [14] and blog posts [15]. Disciplined Agile™ (DA) [16] is a framework supporting agile and lean ways of work. The outcome is focused on solutions rather than on software only. It contains different blocks like Disciplined Agile Delivery

(DAD) and DAE. The DAE focuses on enterprise aspects like legal and governance. Furthermore, it addresses quality in the context of software development and technical debt via the DAD process goals.

3 The Architecture and Characteristics of EFIS

Figure 1 shows an overview of the EFIS framework and the interaction of its individual building blocks. Internal and external regulations (right-hand side) interact with a particular organization within the enterprise (Organization@Enterprise, center). The latter's autonomous value streams implement the business domain-specific regulations and relevant standards the organization is accountable for. Stakeholders contribute knowledge and tool libraries for their domains (bottom right), serving as means of governance interaction between the organization and the enterprise. Each value stream instantiates the relevant library artifacts and enhances them if needed through its contributions via the *Scale* pillar (bottom left).

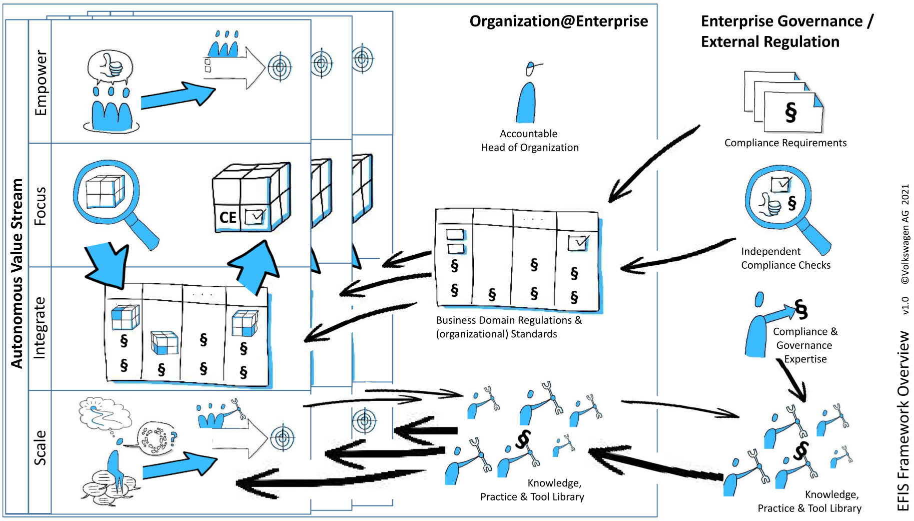

Organization@Enterprise
Enterprise Governance / External Regulation
Autonomous Value Stream
Empower
Focus
Integrate
Scale
CE
§
Accountable Head of Organization
Compliance Requirements
Independent Compliance Checks
Business Domain Regulations & (organizational) Standards
Compliance & Governance Expertise
Knowledge, Practice & Tool Library
Knowledge, Practice & Tool Library
EFIS Framework Overview v1.0 ©Volkswagen AG 2021

Fig. 1. The EFIS framework for autonomous value streams within an enterprise.

The operational core is the instantiation of the *Focus* and *Integration* pillars by identifying the specific product and service quality risks and integrate the related mitigation actions into the value streams' committed set of regulation and standard requirements. Once the requirements are implemented and validated systematically they assure compliant delivery outcomes. The *Empower* pillar (top left) continuously improves teamwork quality, for more mastery which leads to more autonomy, limiting the need for regular team supervision and evaluation.

Openness by design is a cross-cutting aspect for all pillars, and therefore not explicitly modelled. It is achieved by reducing the recommended practices to a minimum. This reduces potential conflicts with other practices and methods that can be included.

Empower. Empowerment of product teams and their organization is achieved through systematic team development that leads to mastery, which is the prerequisite for letting them take over responsibility for their actions in an autonomous way. This, in turn, is indispensable for governing the accountability for any shipped deliverables. Empowered teams can build and improve their delivery procedures and processes independently for fast and innovative solutions. The enabling building block for systematic team empowerment is aTWQ (agile Team Work Quality) as introduced in [17].

Focus. Focus shall be set on each product/service by handling their specific risks for high quality deliveries. Each product or service comes with its business chances that also imply risks that need to be continuously investigated and updated. Systematic risk mitigation actions need to be derived and the effectiveness monitored. At the same time, the organization has to remain open for new innovative products and improvements and associated new risks. EFIS provides the building block PQR (Product Quality Risk) as introduced in [18] to address keeping focus.

Integrate. Integration of processes by interface-driven flows ensure that business domain-specific regulation and governance requirements are implemented for reliable value streams. Value streams need to be identified, including their interfaces and hand-over points. Regulation requirements have to be identified and derived for these value streams and their outcomes, and mapped to the hand-over points. Optionally, organizational intellectual property artifacts related to the value stream can be added to the hand-over points to ensure the property exploration within the value stream processing. Controls associated with the hand-over points enable compliance checks. One assigned individual assures the accountability for the implementation and the compliance governance of the value stream. The EFIS building block to instantiate systematic process integration is LoD (Level of Done) as introduced in [19].

Scale. Scaling of knowledge beyond individual experts and teams is achieved through encouraging knowledge self-services for organizational learning through a *prosumer* (producer and consumer) principle. Learning from self-services to become more mature within the business, product or service domain is encouraged. As are the sharing of any team learnings with others by building new knowledge self-services and updating existing ones. EFIS adopts the SSK (Self-Service Kit) approach as introduced in [20].

The EFIS framework establishes accountability: mature teams master their deliveries, hence they can take responsibility for their actions which enables autonomy.

4 Leveraging Compliance Governance with EFIS

In organizations, development and delivery processes are confronted with a growing number of regulation requirements. To avoid process complexity becoming ever larger, product- and service-related risks can be integrated into corporate governance. In an agile organization, this is feasible since the product teams are responsible for both the process and the product compliance. In EFIS, we integrated guidance and support for this incorporation process through the Product Quality Risk (PQR) building block [18].

EFIS builds on the shared responsibility commitment between the enterprise governance and the local organizational governance for example of a subsidiary or unit. The enterprise governance is able to delegate risk management to the local organization within a shared responsibility approach – however, the local organization risks are still part of the enterprise risk. With this approach, the process and procedure complexity can be reduced by focusing to the explicit process demands for compliance for the specific product – complex one size fit all processes are simplified to the specific products.

All procedure- and process-related compliance aspects are guided by the LoD building block [19], with the topic tasks (t) derived from the regulations. The delivery of relevant internal organizational structures like interfaces and handover-points are modeled through the number of LoD levels. The LoD incorporates the product-specific PQR mitigation actions. The LoD and PQR together make up the core of the lean compliance approach, as depicted in Fig. 2.

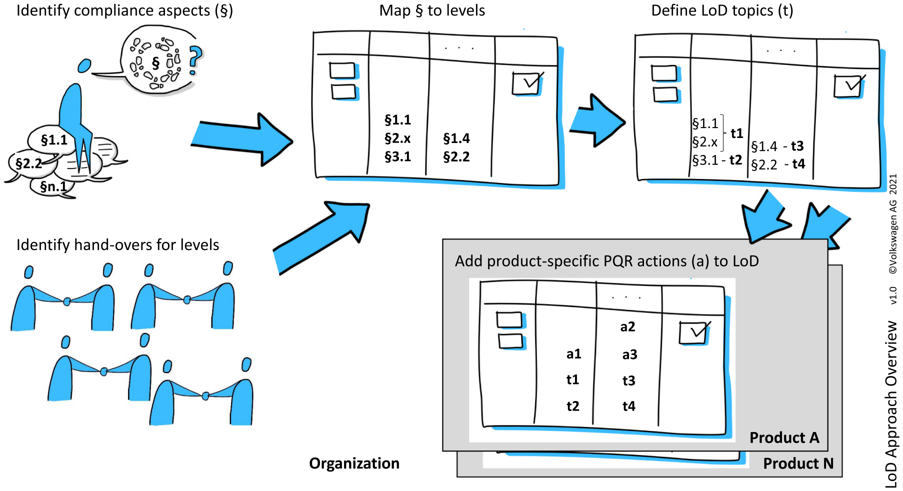


Fig. 2. A domain-specific LoD and its product instantiation.

To reduce the amount and efforts for compliance checks, the organizations' product teams have to be enabled to build compliant deliverables (entire products or services or parts of them as parts of their solutions) and assure their compliance continuously. To realize this, the teamwork quality is key: a more mature teamwork leads to better outcomes, as well as increased performance and quality of deliverables. The team maturity and mastery grows with the teams' skills and capabilities, and the organization can trust and rely on the shared responsibility principle. A highly matured team can master their products and services and work autonomously. To make team maturity transparent and help improve it, the aTWQ approach [17] is part of the empowerment pillar of the EFIS framework.

Scaling individuals' and teams' knowledge within large-scale organizations requires encouraging employees, especially experts, to share their knowledge. In large organizations, however, not all employees know each other. Furthermore, they need to synchronize information and knowledge because the teams' product and service life-cycles are mostly independent of each other. This leads to the demand of providing expert knowledge independently of the experts' current availability. In the EFIS framework, this was realized through the SSK (Self-Service Kit) approach [20]. The essential idea behind SSKs is to share several proven-in-use practices rather than enforce particular practices as a one-size-fits-all approach. The organization and enterprise can further foster the mindset of knowledge sharing and collaboration by establishing incentives such as gamification concepts [21] for contributing to the creation and continuous improvement of SSKs.

5 Instantiation, Evaluation and Improvement

The Volkswagen AG has instantiated and deployed EFIS and its building blocks at different organizations and business areas. The Agile Center of Excellence (ACE) of the Volkswagen Group IT provides it in the form of an *Agile Toolbox* that is available to all Volkswagen Group employees. Additionally the ACE, Test & Quality Assurance (TQA) including Quality innovation NETwork (QiNET) experts are providing coaching and facilitation services.

Some organizational units focus on all pillars, e.g., in the finance domain as well as smaller product delivery organizations such as a cloud service. Other units have been adopting selected pillars to address specific product domain demands in combination with other lean and agile methods. This is possible thanks to EFIS' modular building block architecture. Domain-specific SSK frameworks have been built to scale special matter knowledge to decentralized expert competence fields. In the Spotify model, this corresponds to a guild orchestrated by a squad of experts. However, while Spotify takes a rather structural and organizational approach, EFIS relies on a more operational and operations structuring one. Both can be fitted together, as shown in the example in Fig. 3, visualizing the instantiation adopted for the transversal expert areas like ACE, Machine Learning and Blockchain which are organized in line and virtual organizations. In this example, the Tribe Lead is accountable for the compliance of the organization and owner of the LoD. Furthermore, the Tribe Lead is interested in the continuous improvement of the organization and accountable for the SSKs. The Squads and Chapters have built and maintain both the LoD and the SSKs. They are responsible for fit for purpose in their product or service domain and the operational instantiation. The Squad Leads are accountable for the adequate instantiation of the PQR and aTWQ approach in the teams. The teams are responsible for instantiating and maintaining the EFIS artifacts.

During the EFIS evaluation period, the authors accompanied EFIS instantiations in the IT domain of finance and automotive. The acceptance and added value of EFIS is indicated by the feedbacks of people for example to SSKs and contributions to the enhancement of the EFIS framework like the LoD enhancement with LoD layers by Audi AG and Volkswagen Financial Services AG. Additionally, explicit coaching and facilitation demands are requested to the ACE. New insights and learnings were used to enhance the SSKs and delivery kit of the EFIS framework. Furthermore, the EFIS framework was introduced via the Agile Community of the Volkswagen AG to the community as the last step of the initial evaluation. To improve the EFIS framework continuously, feedback is collected wherever possible. Currently, the LoD layer approach in the context of shared responsibility with different governance units is under development and evaluation. For the teamwork quality pillar, the refinement of aTWQ is in progress. Due to complex IT projects, the teams need technical skills to build high quality products and services. This is addressed by team maturity for specific technologies like cloud computing.

In terms of limitations of this work, the currently known EFIS instantiations are located on different sites and legal entities in Germany. Therefore, there is a lack of information about the application of the EFIS framework in other geographical areas, especially non-European ones. However, some of the investigated teams and organizations in the German sites are highly diverse and international. Furthermore, the SSK approach fostering the autonomous instantiation of the framework and its individual building blocks limits the completeness of measurements and observations related to the actual spread and adoption levels across the entire enterprise group.

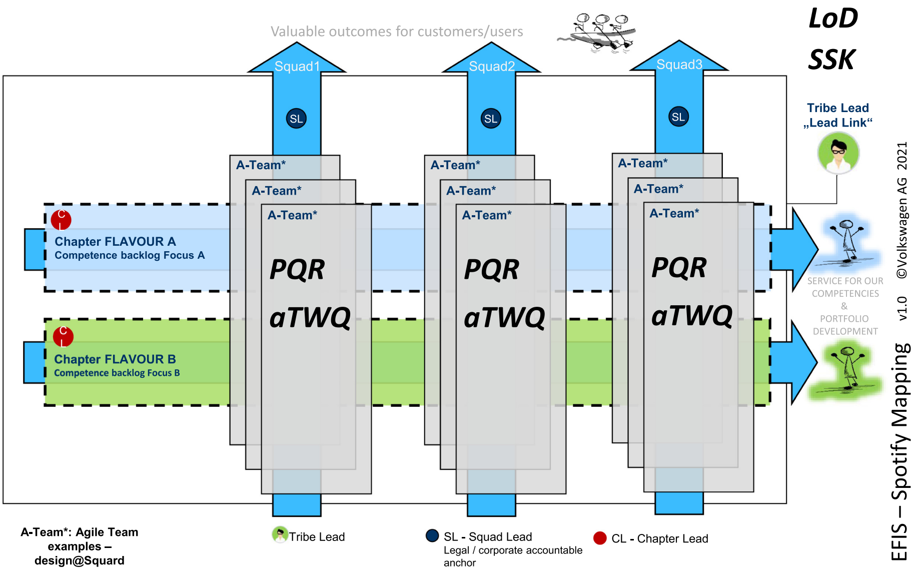

Valuable outcomes for customers/users
LoD
SSK
Squad1
Squad2
Squad3
SL
SL
SL
Tribe Lead
„Lead Link“
A-Team*
A-Team*
A-Team*
PQR
aTWQ
PQR
aTWQ
PQR
aTWQ
CL
Chapter FLAVOUR A
Competence backlog Focus A
CL
Chapter FLAVOUR B
Competence backlog Focus B
SERVICE FOR OUR COMPETENCIES & PORTFOLIO DEVELOPMENT
EFIS – Spotify Mapping v1.0 ©Volkswagen AG 2021
A-Team*: Agile Team examples – design@Squad
Tribe Lead
SL - Squad Lead
Legal / corporate accountable anchor
CL - Chapter Lead

Fig. 3. Mapping of the EFIS practices to a Spotify model oriented organization.

6 Discussion and Limitations

EFIS is a methodology with a supporting set of tools. As SAFe® or Scrum call itself a framework the authors decide to use the term framework, too – to indicate that EFIS can be used stand-alone and as overlay framework on the same level as other established frameworks. An added value for product teams to the generic description of compliance integration like in SAFe® [1, 22] is that with the LoD explicit all relevant regulation and compliance requirements to the specific product are made transparent. Furthermore, all product relevant quality risks are explicit handled. Both together can be used to ensure the product and process compliance of deliveries explicit in the context and responsibility of the product team. Depending on the compliance requirements the teams can do checks manually or automated (e.g. integrated into a continuous delivery chain [23]).

As a limitation of the evaluation the instantiation in the different legal entities is that in all cases the IT applies EFIS – however this indicates that EFIS is applicable to IT organizations and also can be applied outside of the Volkswagen Group. Furthermore, no large metric set about efficiency is established for systematic monitoring and enhancement – only downloads or page views of EFIS SSKs are measured. The currently feedback driven development has to be developed to a more objective indicator and metric driven set. An idea is to use the aTWQ maturity of teams as an long-term indicator of evolvement.

7 Conclusion and Outlook

The EFIS framework proposed in this article provides a way to build a lean and agile environment in large-scale organizations and different domains like automotive IT or finance. Its open, modular design makes it combinable with other lean and agile practices and approaches. In particular, EFIS addresses the needs of large enterprises for systematic team development to facilitate autonomy through mastery. It achieves this through product-specific quality risk management for continuous high quality value delivery, as well as process integration for establishing delivery chains under the enterprise-compliance-governance conditions. Its fundamental underlying lever is knowledge scaling within the entire organization for continuous improvement.

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

- The EFIS framework focusses on domain-specific governance in a lean and agile way, with the LoD to ensure compliance requirements and aTWQ for fostering team autonomy through mastery.
- The EFIS framework has a simple, modular structure based on four pillars and minimizes the amount of methods and practices. This is considered a key success factor for the creation and adoption of specific organizational instantiations.
- The EFIS framework can be combined with other established practices and frameworks like SAFe® to complement them and/or to leverage transitions.

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

- Identification of the gap of systematic governance and quality management by the established lean and agile practices in the context of large enterprises.
- Identification of a way to handle the accountability required by regulations by proposing a chain of accountability, responsibility, mastery and autonomy in large enterprises.

– Demonstration that framework openness and modularity contribute to a holistic yet gradual adoption of agile and lean practices and mindset at a large scale.

Future research will address specific demands of various business domains. Furthermore, specific governance and compliance requirements have been selected for developing lean instantiation approaches and scalable patterns for the affected agile organizations within the enterprise. Additionally, systematic measures have to be established to indicate the effectiveness of the application of the EFIS framework.

References

1. SAFe®. https://www.scaledagileframework.com/. Accessed 09 July 2021
2. LeSS. https://less.works/less/framework/index. Accessed 09 July 2021
3. Nexus. https://www.scrum.org/resources/nexus-guide. Accessed 09 July 2021
4. Scaling Agile@Spotify. https://blog.crisp.se/wp-content/uploads/2012/11/SpotifyScaling.pdf. Accessed 09 July 2021
5. Fitzgerald, B., Stol, K.J., O'Sullivan, R., O'Brien, D.: Scaling agile methods to regulated environments: an industry case study. In 2013 35th International Conference on Software Engineering (ICSE), pp. 863–872. IEEE (2013)
6. Hanssen, G.K., Stålhane, T., Myklebust, T.: SafeScrum®-Agile Development of Safety-Critical Software. Springer, Heidelberg (2018). https://doi.org/10.1007/978-3-319-99334-8
7. Pink, D.H.: Drive: The Surprising Truth About What Motivates Us. Penguin, New York (2011)
8. Hevner, A.R.: A three cycle view of design science research. Scand. J. Inf. Syst. **19**(2), 4 (2007)
9. SAFe® Core Values. https://www.scaledagileframework.com/safe-core-values/. Accessed 09 July 2021
10. SAFe® Continuous Learning Culture. https://www.scaledagileframework.com/continuous-learning-culture/. Accessed 15 July 2021
11. LeSS Organizational Structure. https://less.works/less/structure/organizational-structure. Accessed 15 July 2021
12. The Scrum@Scale Guide. https://www.scrumatscale.com/wp-content/uploads/2020/12/official-scrum-at-scale-guide.pdf. Accessed 15 July 2021
13. Scrum. https://www.scrumguides.org/. Accessed 15 July 2021
14. RAGE. https://www.cprime.com/rage/. Accessed 09 July 2021
15. Recipes for Agile Governance in the Enterprise. https://www.cprime.com/resources/blog/category/agile-articles/. Accessed 23 Apr 2021
16. Disciplined Agile™ (DA). https://www.disciplinedagileconsortium.org/. Accessed 09 July 2021
17. Poth, A., Kottke, M., Riel, A.: Evaluation of agile team work quality. In: Paasivaara, M., Kruchten, P. (eds.) XP 2020. LNBIP, vol. 396, pp. 101–110. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-58858-8_11
18. Poth, A., Riel, A.: Quality requirements elicitation by ideation of product quality risks with design thinking. In: 2020 IEEE 28th International Requirements Engineering Conference (RE), pp. 238–249. IEEE (2020)
19. Poth, A., Jacobsen, J., Riel, A.: Systematic agile development in regulated environments. 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. 191–202. Springer International Publishing, Cham (2020). https://doi.org/10.1007/978-3-030-56441-4_14

20. Poth, A., Kottke, M., Riel, A.: The implementation of a digital service approach to fostering team autonomy, distant collaboration, and knowledge scaling in large enterprises. J. Hum. Syst. Manag. **39**(4), 573–588 (2020). https://doi.org/10.3233/HSM-201049
21. Festinger, L.: A theory of social comparison processes. Hum. Relat. **7**(2), 117–140 (1954)
22. SAFe®. Achieving Regulatory and Industry Standards Compliance with SAFe. https://www.scaledagileframework.com/achieving-regulatory-and-industry-standards-compliance-with-safe/. Accessed 14 July 2021
23. Kellogg, M., Schäf, M., Tasiran, S., Ernst, M.D.: Continuous compliance. In: 2020 35th IEEE/ACM International Conference on Automated Software Engineering (ASE), pp. 511–523. IEEE (2020)

Managing Dependencies in Large-Scale Agile

Henrik Vedal[1], Viktoria Stray[1,2](✉), Marthe Berntzen[1], and Nils Brede Moe[2]

[1] Department of Informatics, University of Oslo, Oslo, Norway
{henrikav,stray,marthenb}@ifi.uio.no
[2] SINTEF Digital, Trondheim, Norway
{viktoria.stray,nils.b.moe}@sintef.no

Abstract. Delivering results iteratively and frequently in large-scale agile requires efficient management of dependencies. We conducted semi-structured interviews and virtual observations in a large-scale project during the Covid-19 pandemic to better understand large-scale dependency management. All employees in the case were working from home. During our data collection and analysis, we identified 22 coordination mechanisms. These mechanisms could be categorized as synchronization activities, boundary-spanning activities and artifacts, and coordinator roles. By using a dependency taxonomy, we analyzed how the mechanisms managed five different types of dependencies. We discuss three essential mechanisms for coordination in our case. First, setting Objectives and Key Results (OKRs) in regular workshops increased transparency and predictability across teams. Second, ad-hoc communication, mainly happening on Slack because of the distributed setting, was essential in managing dependencies. Third, the Product Owner was a coordinator role that managed both inter-team and intra-team dependencies.

Keywords: Product Owner · OKR · Slack · Distributed teamwork

1 Introduction

In large-scale agile software development, teams are surrounded by a larger development context that is often characterized by a high number of dependencies [1]. Teams, therefore, need to understand dependencies within their team as well as to other teams, and to understand how to efficiently manage, or coordinate, these dependencies [2]. A dependency occurs when the progress of one activity, such as a development task, is dependent on the output of a previous activity [3,4]. The more dependencies, the greater the coordination effort is required. Additionally, agile development is the emergence of tasks and work structures during the project [5], which implies that new dependencies will emerge during the development process. Therefore, large-scale agile is characterized by a high number of dependencies that is difficult to plan for up-front, and coordination has been identified as a top challenge to successful large-scale agile [6,7].

P. Gregory and P. Kruchten (Eds.): XP 2021 Workshops, LNBIP 426, pp. 52–61, 2021.
https://doi.org/10.1007/978-3-030-88583-0_6

Table 1. Table of coordination strategy components [8]

Distinct component	Component	Definition
Synchronization	Synchronization activity	Activities performed by all team members simultaneously that promote a common understanding of the task, process, and or expertise of other team members
	Synchronization artefact	An artefact generated during synchronization activities. The nature of the artefact may be visible to the whole team at a glance or largely invisible but available. An artefact can be physical or virtual, temporary or permanent
Structure	Proximity	This is the physical closeness of individual team members. Adjacent desks provide the highest level of proximity
	Availability	Team members are continually present and able to respond to requests for assistance or information
	Substitutability	Team members are able to perform the work of another to maintain time schedules
Boundary spanning	Boundary spanning activity	Activities (team or individual) performed to elicit assistance or information from some unit or organization external to the project
	Boundary spanning artefact	An artefact produced to enable coordination beyond the team and project boundaries. The nature of the artefact may be visible to the whole team at a glance or largely invisible but available. An artefact can be physical or virtual, temporary or permanent
	Coordinator role	A role taken by a project team member specifically to support interaction with people who are not part of the project team but who provide resources or information to the project

Research within large-scale has addressed coordination at the organizational, project, inter-team and team level in both distributed and co-located settings [1,8–10]. While research-based knowledge on coordination in large-scale agile is expanding, there are still unresolved questions, such as which coordination mechanisms used in large-scale agile are more effective. Guided by the need for more knowledge on coordination in large-scale agile, we address the following research question: *How are dependencies managed in large-scale agile projects?* To address this question, we report on a case study conducted in a distributed development team in a large-scale organization.

1.1 A Framework for Coordination in Agile Teams

In this study, we rely on a theory of coordination for agile teams developed by Strode and colleagues [8]. The theory is relevant in large-scale contexts because it takes into account that organizational structure, project complexity, and uncertainty influence coordination [8]. We chose this theory as a lens for investigating large-scale coordination because it captures both explicit coordination (e.g., an insight report) and implicit forms of coordination (such as knowledge sharing) [1,8].

Coordination mechanisms are specific activities and artifacts aimed at addressing dependencies of three types [3]: 1) Knowledge dependencies is when a form of information is needed for a project to progress and consist of four sub-categories: expertise, requirement, task allocation, and historical. 2) Process dependencies are defined through two categories, activity and business process, which entails when a task must be completed before another task can be initiated. 3) Resource dependencies are composed of entity and technical, which is when an object is needed for a project to progress. For example, entity dependency is when a person is not available and this affects project progress [3].

The theoretical framework proposes three categories of coordination mechanisms to manage these dependencies (see Table 1): Synchronization mechanism such as daily stand-up meetings and product backlog; structure mechanisms referring to the proximity, availability, and substitutability of team members; and boundary spanning mechanisms that connect interdependent teams [3,8].

The theory further proposes that agile coordination mechanisms lead to coordination effectiveness, where agile team members have a shared understanding of their goals and priorities as well as how each team member's tasks fit together, as well as the necessary tools and artefacts available at the right time and place, thereby being able to sufficiently manage their dependencies [8]. The original framework included typical agile coordination mechanisms, such as daily stand-up meetings and the product backlog [3,8]. However, in large-scale settings, it is common to also use other project management mechanisms [1], including goal-setting frameworks [10]. One such framework that we will focus on in this study is the Objectives and Key results-framework, described next.

1.2 Objectives and Key Results

Objectives and key results (OKR) is a goal-setting framework to define a certain set of objectives in an organization and measure its progress. Instead of spending months setting long-term goals, OKR is designed to help organizations achieve their business goals much quicker in a structured manner. It is defined as "a critical thinking framework and ongoing discipline that seeks to ensure employees work together, focusing their efforts to make measurable contributions that drive the company forward" [11, p. 6]. An objective describes in short terms what the team wants to achieve, while key results allow the team to measure their progress and show when the objective has been reached.

The OKR-framework focuses on balancing business value and measureability [11]. This may explain why large-scale agile companies choose to adopt it. Another attractive feature of OKR that is compatible with agile is the emphasis on broad participation and collaboration across organizational levels [11], right down to the development teams.

2 Research Method

We chose to conduct a case study because it is an empirical research method for investigating contemporary phenomena and because it is particularly fit when the boundaries between context and phenomenons are ambiguous or not apparent [12]. Our case is an agency within a sizeable Norwegian municipality with approximately 50,000 employees distributed among 50 organizational units. Established as a project in 2017, but later organized as an agency in early 2020, the case comprises six departments and seven product areas, consisting of 11 permanent and three temporary teams structured with people from multiple departments (Fig. 1). These cross-functional agile teams deliver solutions such as web solutions, mobile solutions, document-handling solutions, and business systems. The work entails connecting existing systems in the municipality to create a shared service platform for agencies and businesses. Considering the amount of data the municipality holds, the objective is to facilitate the creation of high-quality, valuable services for all citizens of the municipality.

Our primary data collection was from Team Alpha. Their goal was to create a platform to facilitate easy access and sharing of data within agencies in the municipality and ensure it is being put to use. This included making intuitive and secure solutions, good documentation, and ultimately enabling the citizens to have access to better solutions. The team had ten permanent members and one part-time member (a UX designer). All permanent team members were interviewed in December 2020. See Table 2 for a description of the different roles. We interviewed one team lead, one tech lead, one UX designer, four back-end developers, two front-end developers, and one data scientist. The video conferencing tool Zoom was utilized, allowing easy access to record the interviews with

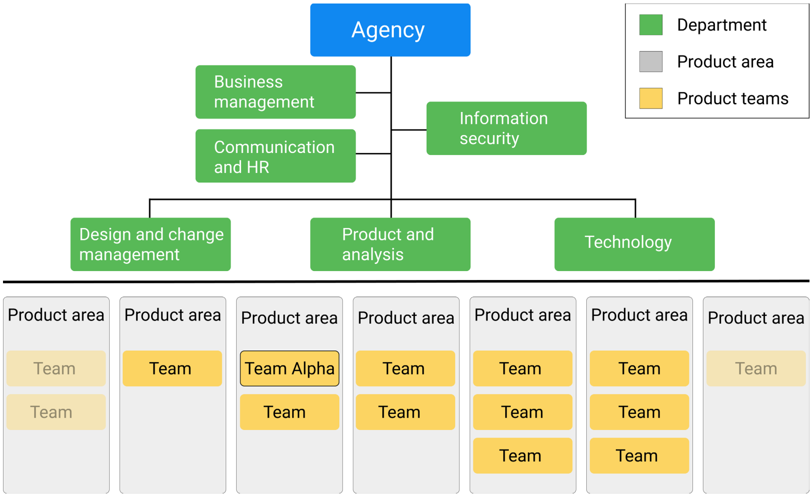


Fig. 1. The large-scale set-up.

permission from the interviewee. The average interview length was 48 min, and all interviews were transcribed.

We analysed the interview transcripts by systematically coding them in Nvivo 12. One analytical strategy proposed by Yin [12] is the reliance on a theoretical proposition, and we chose to guide the data analysis by using the coordination framework outlined above [8]. Following this framework, the analysis was organized and helped us point out relevant contextual conditions [12].

Table 2. Roles in team Alpha

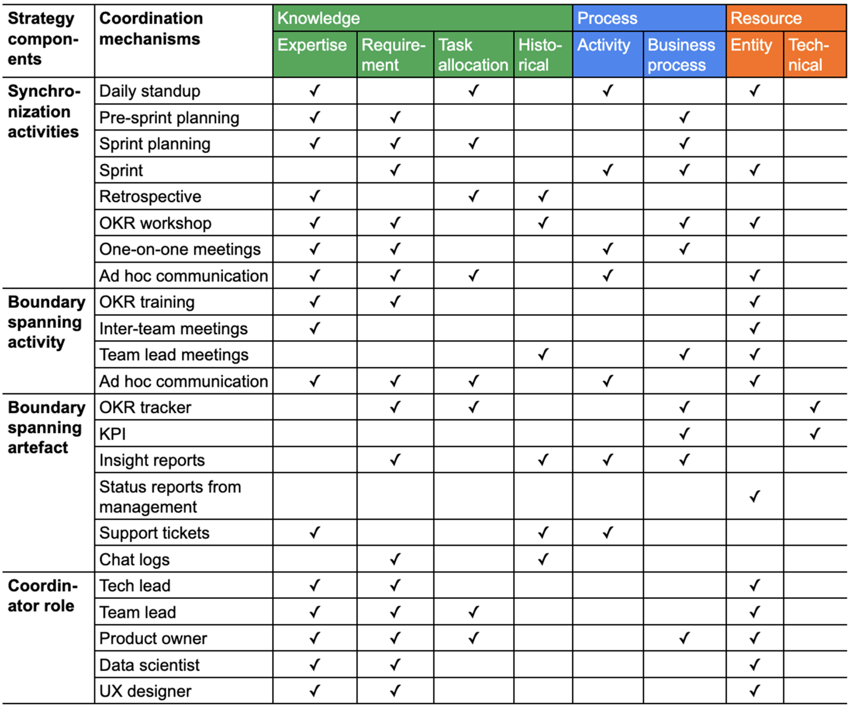
Role	Members	Description
Team lead	1	Ensures that the team is moving in the right direction, communicating company goals, and facilitating a good flow of information
Tech lead	1	Also referred to as the architect, performs development work, coordination, and provides technical guidance for the team members
UX designer	2	Works with illustration, design, and collection of data and insight from other teams and agencies
Back-end	4	Management and operation of the existing systems, as well as the continuous implementation of new server-side features
Front-end	2	Development of new client-side functionality, creating a user interface and coordination with designers
Data scientist	1	Acquires requirements, use cases, and proof of concept for new features to be built

3 Results

We used the dependency framework [8] to categorize coordination mechanisms and how they managed the different dependencies. Figure 2 gives an overview of the identified coordination mechanisms and how they address relevant dependencies. In addition to the components shown in the figure, we also identified 11 synchronization artifacts, such as Kanban boards showing the status of the different tasks, and Github pages for documentation. In the following, we report on three coordination mechanisms that addressed the most dependencies (five or more) since these indicate to be the most important. These were OKR workshop, ad-hoc communication and Product Owner (PO).

3.1 OKR Workshop

Working towards the same goals was identified as a critical success factor. To achieve this, the project relied on the OKR framework. Every quarter the teams



Strategy components	Coordination mechanisms	Knowledge				Process		Resource	
		Expertise	Requirement	Task allocation	Historical	Activity	Business process	Entity	Technical
Synchronization activities	Daily standup	✓		✓		✓		✓	
	Pre-sprint planning	✓	✓				✓		
	Sprint planning	✓	✓	✓			✓		
	Sprint		✓			✓	✓	✓	
	Retrospective	✓		✓	✓				
	OKR workshop	✓	✓		✓		✓	✓	
	One-on-one meetings	✓	✓			✓	✓		
	Ad hoc communication	✓	✓	✓		✓		✓	
Boundary spanning activity	OKR training	✓	✓					✓	
	Inter-team meetings	✓						✓	
	Team lead meetings				✓		✓	✓	
	Ad hoc communication	✓	✓	✓		✓		✓	
Boundary spanning artefact	OKR tracker		✓	✓			✓		✓
	KPI						✓		✓
	Insight reports		✓		✓	✓	✓		
	Status reports from management							✓	
	Support tickets	✓			✓	✓			
	Chat logs		✓		✓				
Coordinator role	Tech lead	✓	✓					✓	
	Team lead	✓	✓	✓				✓	
	Product owner	✓	✓	✓			✓	✓	
	Data scientist	✓	✓					✓	
	UX designer	✓	✓					✓	

Fig. 2. Coordination mechanisms and dependencies identified.

arranged OKR workshops to set the direction and linking to the overall objectives of the project to the teams. Each team was encouraged to look at other teams' OKRs to understand the dependencies between teams. Team Alpha's OKR meeting handled internal team dependencies such as expertise, requirement, historical, business process, and entity.

Further, the OKR workshop managed expertise and requirement dependencies because they relied on very specific knowledge to create optimal OKRs. The workshop also managed entity dependencies as many individuals on the team provided valuable knowledge to ensure the quality of the OKRs. The team members stated how OKRs provided increased transparency, predictability, shared goals and an increased sense of ownership to what was produced in the project. The agency also utilized an OKR tracker, a tool which allowed any team to view the progress of other teams. One of the hardest parts about using OKR that was stated by several team members, was quantifying objectives through key results and the corresponding choice of words. A member of team Alpha stated: "*I think OKR is difficult. It is useful to maintain focus, but it is hard to create good, measurable key results which make sense.*" This further emphasizes the need for this workshop to manage the dependencies, as they are reliant on it to improve their collective understanding and set better OKRs.

3.2 Ad Hoc Communication

Besides regular inter-team and intra-teams meetings, there was a substantial amount of communication performed ad hoc, mainly using Slack. The ad-hoc communication on Slack managed expertise (team members reached out to other experts in the project), requirements (a team member located specific product related domain knowledge), task allocation (discussions led to emerging tasks) and activity dependencies (a team member needed information to continue work).

The Slack infrastructure provided public slack channels, dedicated team channels and private messages. Private messages was used both internally in the team and externally to communicate with others in the large-scale project. While much information was sent as private messages, the project members were encouraged to ask questions in open channels. When team members were unsure about details of the domain or technology, they found it easy to reach out on Slack to locate this information. The tech lead explained the following: *"Our team have a lot of experience with cloud technology and other project members often ask for assistance. We have similarly reached out to other teams that have specific knowledge which might be relevant for us".* This spontaneous communication often led to unscheduled meetings and positively benefitted the participants. The communication also involved agreeing on pair programming and discussing debugging. Because all project members were distributed, this coordination mechanism was probably more evident as the majority of communication during work hours was digital.

3.3 Product Owner

The PO was a coordinator role that managed a total of five dependencies. The role managed dependencies of types expertise, requirement, task-allocation, business process, and entity, as shown in Fig. 2. The PO communicated stakeholder interests, checked status, and pointed teams in the right direction. The majority of the work performed by the PO was coordination related to both intra- and inter-team coordination. Working tightly with the team lead and tech lead of Team Alpha, the PO assisted in the discussion of which key results (related to OKR) to prioritize throughout the quarter. This is what ultimately decides many of the tasks which the team works with during any point in time, which in turn manages requirement and task allocation dependency. It was not always evident to the product teams what to prioritize. The PO managed much of the inter-team coordination with the appropriate teams. This allowed the teams to be aligned and focus on their product, enabling autonomy and avoiding potential bottlenecks and misunderstandings.

4 Discussion

Understanding coordination in a distributed environment is critical to project success [13]. We studied how dependencies were managed in a large-scale agile project that was forced to work from home due to Covid-19. We now discuss our research question: *How are dependencies managed in large-scale agile projects?.*

The use of OKRs served as an essential mechanism for managing dependencies because the approach was a foundation for setting direction for all teams. We identified how the OKR workshop managed dependencies such as expertise, requirement, historical, business process, and entity. The key to a successful workshop was having an overview of past decisions and specific knowledge about the product and project goals. In the workshop, team lead and tech lead managed requirement dependencies, which enabled effective prioritization, which is vital in high uncertainty complex projects [8].

The second most important mechanism to dependency management was ad hoc communication, which addressed expertise, requirement, task allocation, activity, and entity. For example, Team alpha often agreed to do pair programming ad hoc, which was positive, since one of the main challenges of remote pair programming is the initiation of pairing [14].

All scheduled and unscheduled communication was performed digitally. However, a high number of Slack channels resulted in challenges of getting an overview of what was going on in the project. Our findings are in accordance with [15], which found that coordination challenges are evident in distributed teams. Our results further showed that it was hard to balance ad hoc communication on Slack and scheduled meetings. We found that some discussions could go on for too long on Slack which created misunderstandings, instead of organizing a meeting to discuss the dependency and resolve the blocking of progress. Our findings are consistent with the findings of [13], which showed that a lack of guidelines when using Slack resulted in coordination being confusing and frustrating for team members. Another challenge with the use of Slack was that it was expected that the project members answered reasonably quickly, which some interviewees said led to increased interruptions and context switching in the distributed teams.

The third most important mechanism for managing dependencies was the PO role. The PO managed knowledge, process, and resource dependencies. Our results confirm previous research, that the PO role is important in large-scale agile for managing dependencies between and within the team [16–18]. In our case, the PO worked in close collaboration with the team leader and tech lead, thus managing process-related and technical dependencies. In accordance with the findings of Bass [17], we found that the PO performed a wide set of different functions. Our findings are also in line with Remta et al. [18]. In their study of POs in a company that implemented the Scaled Agile Framework, they found that the PO role entails responsibilities such as being a gatekeeper, motivating, communicating and prioritising [18], addressing five types of dependencies.

5 Conclusion and Future Work

In conclusion, our study suggests that by discussing OKRs, the teams manage dependencies both within the teams and also inter-team dependencies. We found that ad-hoc communication mostly happened on Slack and that this communication made team members locate expertise in the large-scale project as well as discussing requirements, task-allocation and activity dependencies. The PO played an important role because it managed five different types of dependencies. In this large-scale project, there was no dedicated role focusing on OKRs, so discussions about them in the teams took time during meetings and required much facilitation. Our findings indicate that large-scale projects would benefit from having a dedicated "OKR master" to facilitate and follow up the OKR process. Future work should further explore how OKRs can be used to align teams in a large-scale distributed set-up.

References

1. Dingsøyr, T., Moe, N.B., Seim, E.A.: Coordinating knowledge work in multiteam programs: findings from a large-scale agile development program. Proj. Manag. J. **49**(6), 64–77 (2018)
2. Malone, T.W., Crowston, K.: The interdisciplinary study of coordination. ACM Comput. Surv. (CSUR) **26**(1), 87–119 (1994)
3. Strode, D.E.: A dependency taxonomy for agile software development projects. Inf. Syst. Front. **18**(1), 23–46 (2016).
4. Crowston, K., Osborn, C.S.: A coordination theory approach to process description and redesign (1998)
5. Boehm, B., Turner, R.: Management challenges to implementing agile processes in traditional development organizations. IEEE Softw. **22**(5), 30–39 (2005)
6. Bass, J.M.: Future trends in agile at scale: a summary of the 7th international workshop on large-scale agile development. In: Hoda, R. (ed.) XP 2019. LNBIP, vol. 364, pp. 75–80. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-30126-2_9
7. Bass, J.M., Salameh, A.: Agile at scale: a summary of the 8th international workshop on large-scale agile development. In: Agile Processes in Software Engineering and Extreme Programming-Workshops, p. 68 (2020)
8. Strode, D.E., Huff, S.L., Hope, B., Link, S.: Coordination in co-located agile software development projects. J. Syst. Softw. **85**(6), 1222–1238 (2012)
9. Stray, V., Moe, N.B., Mikalsen, M., Hagen, E.: An empirical investigation of pull requests in partially distributed BizDevOps teams. In: The 16th ACM/IEEE International Conference on Global Software Engineering (ICGSE), pp. 110–119 (2021)
10. 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
11. Niven, P.R., Lamorte, B.: Objectives and Key Results: Driving Focus, Alignment, and Engagement with OKRs. Wiley, Hoboken (2016)
12. Yin, R.: Case Study Research and Applications: Design and Methods, 6 edn. SAGE Publications, Upper Saddle River (2017)
13. Stray, V., Moe, N.B.: Understanding coordination in global software engineering: a mixed-methods study on the use of meetings and slack. J. Syst. Softw. **170**, 110717 (2020)

14. Smite, D., Mikalsen, M., Moe, N.B., Stray, V., Klotins, E.: From collaboration to solitude and back: remote pair programming during Covid-19. In: Gregory, P., Lassenius, C., Wang, X., Kruchten, P. (eds.) XP 2021. LNBIP, vol. 419, pp. 3–18. Springer, Cham (2021). https://doi.org/10.1007/978-3-030-78098-2_1
15. Espinosa, J.A., Slaughter, S.A., Kraut, R.E., Herbsleb, J.D.: Team knowledge and coordination in geographically distributed software development. J. Manag. Inf. Syst. **24**(1), 135–169 (2007)
16. Berntzen, M., Moe, N.B., Stray, V.: The product owner in large-scale agile: an empirical study through the lens of relational coordination theory. In: Kruchten, P., Fraser, S., Coallier, F. (eds.) XP 2019. LNBIP, vol. 355, pp. 121–136. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-19034-7_8
17. Bass, J.M.: How product owner teams scale agile methods to large distributed enterprises. Empir. Softw. Eng. **20**(6), 1525–1557 (2015).
18. Remta, D., Doležel, M., Buchalcevová, A.: Exploring the product owner role within safe implementation in a multinational enterprise. In: Paasivaara, M., Kruchten, P. (eds.) XP 2020. LNBIP, vol. 396, pp. 92–100. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-58858-8_10

Summary of First International Workshop on Agile Sustainability

A Summary of the First International Workshop on Agile Sustainability

Juan Garbajosa[1], Coral Calero[2], Jennifer Perez[1], and Agustin Yagüe[1]

[1] Research Center on Software Technologies and Multimedia Systems
for Sustainability (CITSEM), Universidad Politécnica de Madrid, Madrid, Spain
{juan.garbajosa,jenifer.perez,agustin.yague}@upm.es
[2] Institute of Technology and Information Systems, University of Castilla-La Mancha,
Ciudad Real, Spain
Coral.Calero@uclm.es

Abstract. This is a summary of the First International Workshop on Agile Sustainability. The workshop analysed sustainability from the agile perspective through research presentations and collaborative activity. The keynote was focused on considering the world as a customer and the two papers address really interesting and varied ideas about collective intelligence and sustainability and how to increase the sustainability awareness in the Agile community. The collaborative activity analysed the established agile practices from the sustainability point of view.

Keywords: Agile · Sustainability · SDG

1 Introduction

For years, sustainability has been a concern for many communities, and this has been the case for the Agile software development community. The definition of the UN 2030 Agenda for Sustainable Development has provided more concrete dimensions to sustainability since at the heart of the Agenda are the 17 Sustainable Development Goals (SDGs) [7]. Organizations are currently redefining the strategic objectives and processes, so that all these can be better aligned with these 17 SDGs. The three main dimensions of sustainability are Environmental, Social, and Economic [4, 5, 6]. The Environmental dimension is related to the long-term effects that our actions will have on the planet in terms of climate, primary resources, water, pollution, consumption, etc. The Social is concerned with the communities of our society and those factors that affect them in a positive or negative way. And finally, the Economic dimension is focused on assets, capital, added value, and return on investment.

Society, industry, and research community is being aware of sustainability, even refining this three dimensions in other more specific [3]. Therefore, the agile community cannot look the other way and it is required to study Agile and Sustainability in depth and from a new and more fresh perspective, seeking a mutual benefit of both Agile and sustainability. It should not be forgotten that Agile was born on top of a set of values, principles, and practices and, in this respect, Agile is different from other development paradigms. The relationship and impact that the Agile paradigm and the agile software development approach have on the sustainability dimensions and on the SDGs, require to be thoroughly considered. Issues such as resource optimization (human, technical, ...), perdurability, and energy efficiency both from the process and product perspectives are relevant.

Purvis et al. [8] stated the three-pillar conception of sustainability comprising social, economical, and environmental are still far from clear. The Agile Manifesto [2] could be a useful guide to build and to trace the footprint of products. How to map agile values and principles is not new and some research has been conducted before. However, in this workshop, we would like to take a step forward by researching on different approaches and mapping the agile practices documented by the Agile Alliance [1] with sustainability to create adoption roadmaps to build sustainable software. To that end, the workshop was not only based on research presentations about sustainability and agile, but also it provided a collaborative activity among attendants.

2 Workshop Development

As it has been mentioned before, the workshop was divided in two sessions: the session of research presentations and the session of collaborative activity. The first one comprised a keynote and two additional papers. The keynote "*Agile Sustainability: What if the world is our customer?*" by Jutta Eckstein presented a holistic perspective on who our customer is (or could be) and how this can change our actions from the sustainability point of view. She shared both ideas about possibilities and examples from companies who make already attempts in implementing this holistic perspective. It was an open-minded session on how the agile community can expand their responsibilities.

The first presentation was "*How Collective Intelligence Can Gear Agility with Sustainability*". It was presented by Juan Ochoa-Zambrano. He talked about how agile methodologies and sustainability goals are somehow aligned, and that this alignment can be advantageously used to implement new transformation approaches to implement a more effective adoption of both agile and sustainably goals in organizations. He was focused on "collective intelligence" which has proven to be a very powerful tool to generate solutions to complex problems by combining the diversity of knowledge and skills of different actors into better solutions or processes, which can be extended to wider contexts. In addition, individuals participating in any collaborative process benefit from the level of skills and new knowledge. Specifically, this paper points out how the application of collective intelligence could be applied to support a transformation process to achieve the adoption of the Agile and Sustainability goals.

The second presentation "*Increasing Awareness of Sustainability in the Agile Community: A Focus Group Protocol*" was presented by Claudia Melo. She described her proposal for the focus group protocol to increase sustainability awareness in the Agile community. The protocol was based on the usage of flashcards that allow both being informational and propositional. With the flashcards' focus on active recall, the participants of the focus groups will increase their knowledge on sustainability and, at the same time, explore options inspired by questions.

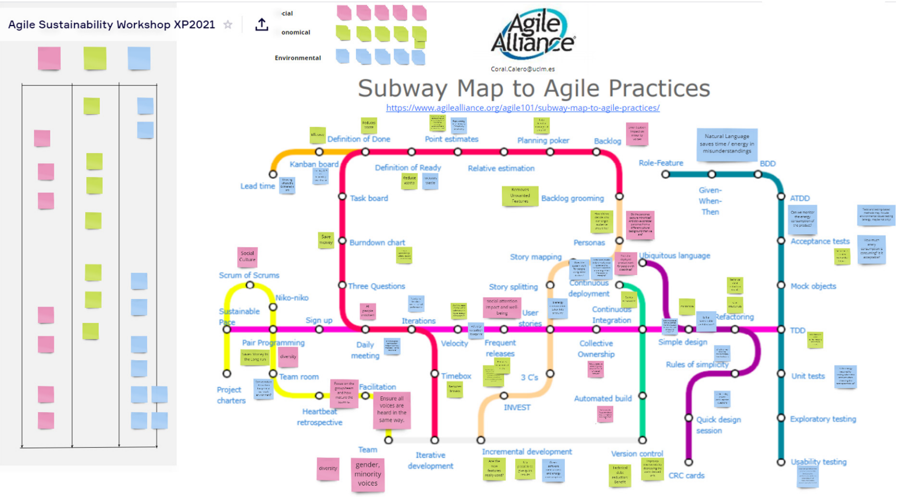


Fig 1. Miro panel created during the workshop.

The second session was a collaborative activity with the main goal of identifying the next steps to address sustainability in agile communities and development, and working on an agile sustainability roadmap. It was based on analysing the established agile practices [1] from the sustainability point of view. To that end, a board with the subway map of Agile practices was used to drive the activity. The practice was conducted as a world cafe[1] having three rounds and each round analyzed a pillar of Agile Sustainability (economical, social, and environmental) in a time slot of 12 minutes. A colour was assigned to each pillar: pink for social, green for economic, and blue for environmental. Then, attendees added colored post-it notes to those practices that considered could have an impact or be impacted by the analyzed pillar. After these three rounds, the last time slot was focused on wrapping up conclusions and determining future works.

To support the world cafe, a Miro board was created and shared with the workshop attendees. Figure 1 shows an overview of the generated board. Pink sticky notes represent those practices that impact the social pillar. Green ones, those practices impacting on the economical pillar and, finally, the environmental pillar is documented by blue sticky notes. In the next section, the main concerns and conclusions are presented.

[1] http://www.theworldcafe.com/key-concepts-resources/world-cafe-method/.

3 Workshop Conclusions and Next Steps

After more than three amazing hours, some interesting results were obtained concerning practices and agile sustainability pillars. Concerning the social pillar, 13 sticky notes were added and can be summarized that most of the practices in the subway map related to *Teams* have a great impact in this pillar. However, some other agile practices like “Personas”, “Collective ownership” or “Daily meeting” were also labeled as important for this pillar. In the case of “Persona”, it was highlighted the relevance of capturing minorities and considering different cultural background.

Evaluating the impact of the agile “Economical” pillar, 24 sticky notes were collected. Most of them were on the topic of “Product Management” and “Product definition” mainly addressing product estimation and backlog refinement. It was also emphasized the relevance of the Scrum practices Timebox and Burndown chart, because they help agile to reduce the waste generated by interruptions and to understand better how value is created.

Finally, the “Environmental pillar” was analyzed based on 21 sticky notes. The mass of the identified practices were in the “Testing” and “DevOps” areas. Considering that both areas are highly dependent on technology where power consumption is critical. It was also underlined the impact of those practices on the area of system design and how to create sustainable architectures. Something that was also identified is that, in most of the cases, the economical and environmental dimensions have an inversely proportional relationship and in terms of sustainability some trade-offs are required.

References

1. Alliance, A.: Subway map to agile practices (2021). https://www.agilealliance.org/agile101/subway-map-to-agile-practices/
2. Beck, K.: Manifesto for agile software development (2001). http://www.agilemanifesto.org/
3. Becker, C., et al: Sustainability design and software: the karlskrona manifesto. In: Proceedings - 2015 IEEE/ACM 37th IEEE International Conference on Software Engineering, ICSE 2015. vol. 2, pp. 467–476 (Aug 2015)
4. Calero, C., Piattini, M.: Puzzling out software sustainability. Sustain. Comput. Informatics Syst. 16, 117–124 (2017). https://doi.org/10.1016/j.suscom.2017.10.011
5. ESCAP, U.: Integrating the three dimensions of sustainable development: A framework and tools. Greening of economic growth series, UN, New York (2015)
6. Nations, U.: Report of the world commission on environment and development: note, p. 374 (Aug 1987)
7. Nations, U.: Sustainable development goals (2015). https://sdgs.un.org/es/goals
8. Purvis, B., Mao, Y., Robinson, D.: Three pillars of sustainability: in search of conceptual origins. Sustainability Science (September 2018). https://eprints.whiterose.ac.uk/136715/

How Collective Intelligence Can Gear Agility with Sustainability

Juan Ochoa-Zambrano(✉)

Universidad Peolitécnica de Madrid, 28031 Madrid, Spain
js.ochoa@upm.es

Abstract. Some emergent research works have identified that Agile methodologies and sustainability goals are, somehow, aligned. This alignment can be advantageously used to implement new transformation approaches with the objective of implementing a more effective adoption of both Agile and sustainably goals in organizations. Studies claim that Agile and sustainability can be geared with team collaboration and learning. Collective Intelligence has proven to be a very powerful tool, to generate solutions to complex problems, because it is able to combine the diversity of knowledge and skills of different actors into better solutions or processes, which can be extended to wider contexts. In addition, individuals participating in any collaborative process, benefit at the level of skills and new knowledge. In this article, the application of the concepts of collective intelligence to support a transformation process in which the combined adoption of the Agile and Sustainability goals is described.

Keywords: Collective Intelligence · Agile · Sustainability · Team Diagnostic Survey

1 Introduction

The UN in its Sustainable Development Agenda for 2030, mentions 17 Sustainable Goals [1], those goals are a concern for our society, and, according to the position elaborated in [2], it could happen that the journey to a more sustainable society is performed in companion with Agile. In the software development world, agility was born to face a needed change in the way software was developed. An emerging interest around agility and sustainability is taking place and providing its first results. Agile, for the software world, was established in the Agile Manifesto as a set of values, and principles, that would guide the execution of practices [3]. Melo and Eckstein [4] have elaborated how, from Agile software development principles and practices, sustainability can be comfortably, and even, necessarily considered. It looks like Agile could not ignore sustainability.

It is interesting to see how this situation also stands in the case of Agile and sustainability in other domains, different from software development. In the case of organizational culture values and agility [5], it happens that sustainability comes hand in hand with agility. It is also the case project management in Engineering [6]. After analyzing literature from different domains, even when a more in-depth research could be needed

P. Gregory and P. Kruchten (Eds.): XP 2021 Workshops, LNBIP 426, pp. 69–77, 2021.
https://doi.org/10.1007/978-3-030-88583-0_7

to get more empirical evidence, it looks like agility and sustainability could supplement with each other, one brings/needs the other.

Following [4], values and principles are a gate to sustainability and sustainable development. To understand the relevance of values in Agile, Sommer [7] reports that the adoption of Agile values has a very positive impact in the operation of the transformed to Agile organization. Based on the alignment of Agile and sustainability discussed above in this Section, it could make sense that transformations could consider both Agile and sustainability.

To understand how this transformation, involving Agile and sustainability can be reached, authors in [2, 3, 5, 6], provide us with some results of interest, especial mention to [3], an empirical study. These papers agree that the journey towards Agile and sustainability is supported by a common background, including learning and team collaboration as relevant activities. Learning and collaboration seem to gear agility with sustainability.

Team collaboration and learning are regarded as central issues to Agile. Patterns have been identified, nevertheless, a systemic approach that can be applied to support or get advantage of them [8] does not seem to be feasible, at least in available literature.

Collective Intelligence (CI) is a paradigm that emerges naturally in groups of individuals who collaborate to solve a set of complex tasks or problems; and according to Malone in [9], Collective Intelligence can be defined as the ability of a group of individuals to act in an intelligent way. Collective Intelligence has been used in Agile software development such as by Diegmann and Rosenkranz [10].

Within this current paper we claim that CI can be used to gear Agile and sustainability because its application will help assess and improve team collaboration and learning.

The rest of the paper is as follows, Sect. 2 defines the concepts of Collective Intelligence, Agility, Sustainability, and the relationship between them. Section 3 describes the relationship between Collective Intelligence and Team Learning, providing a background on these subjects. Section 4 is a proposal on how Collective Intelligence could be used to gear Agile with Sustainability. Finally, Sect. 6 include some conclusions and future work.

2 Collective Intelligence, Agility, and Sustainability

Nowadays, ICTs plays an important role in achieving global sustainability goals, reducing not only their own carbon footprint, but also as a tool to find solutions to help reduce the carbon footprint resulting from society's production and consumption [11]. For this, two terms should be considered, sustainable development and sustainable use. On the one hand, sustainable development is based on creating goods and services that are more sustainable during their life cycle, on the other hand, sustainable use is based on creating and promoting sustainable patterns of consumption and production [11].

Therefore, at any stage of software development, we must be able to create solutions with better performance that meet the new global needs of sustainability. In fact, Agile methodologies must begin to consider these new requirements. A first approach is presented by Eckstein and Melo in [4], where they mention that sustainability is directly related to the Agile Manifesto and more specifically to the Agile principles. Taking one of the Agile values, which says "Our highest priority is to satisfy the customer through early and continuous delivery of valuable software", we can realize that nowadays the definition of "valuable software" has evolved to pick up new needs of citizens concerning global sustainability goals [4]. This means that developers in Agile teams will find themselves in continuous learning considering the three pillars of sustainability [4]: economic, environmental, and social.

And at this point, how CI is related with sustainability? Collective Intelligence is not a new term, naturally emerges from individuals working together, allowing to extend the knowledge extracted locally to a global scale, since through it is possible to learn from the experiences and diversity of other groups, which contributes to identify other ways to solve the same problem or conflict from a completely different perspective [9]. Those things that "work~~s~~" can be transmitted from the local to the global context, in order to obtain human systems (societies, governments, organizations) with a set of policies, programs, behaviours, actions that produce good results [12]. Thanks to ICTs, new forms of CI have emerged, allowing communities of individuals to use the new infrastructures to transmit, create or exchange knowledge and resources [13]. In [13] the potential of different CI techniques to address different social problems is shown. Although its adoption has not been as widespread as desired, it is shown that the aggregation of knowledge for decision making for the generation of solutions gives better results than if it were done by a single individual. Collaboration seems to be the right strategy to generate solutions that meet the new global objectives [13]. For example, in [14] the authors apply CI through a Serious Game to help generate solutions in which citizens and students participate. Typically, decisions on what to do in public spaces were made by a small group of authorities, architects, engineers, etc. Through tools such as those adopted in [14], with a bottom-up perspective, citizen participation is promoted while improving students' knowledge concerning sustainability by allowing the transfer of knowledge to real contexts. In [15] a similar idea is presented, proposing a project whose main objective is to use Cultural Heritage to create a collective decision-making network to assist public and private entities and citizens. The reuse of Cultural Heritage has allowed sustainable urban development, mitigation of the adverse effects of climate change, waste reduction and much more. This, added to the new techniques in Collective and Artificial Intelligence, allows the generation of richer information networks that contribute to the creation of solutions with less development time and better resource management [15].

As can be inferred CI, agility, and sustainability, have a lot to do in common and are mainly related to the transfer of knowledge between individuals or diverse groups, therefore directly correlated with team learning, which will be discussed in more depth in the following section.

3 Collective Intelligence and Team Learning

Concerning problem-solving, there are various perspectives, such as technology-centric, human-centric and CI centric, but the collective intelligence-perspective has proven to be extremely useful when solving problems for the welfare of humanity [16]. Complex problem solving in science, engineering and business has become a highly collaborative effort, where the collaboration and expertise of individuals from different disciplines is required. Teams of scientists or engineers collaborate together on projects using their social networks to gather new ideas and feedback [17]. CI will also emerges from groups of individuals collaborating together [18, 19], and according to Barlow in [20] CI is the ability to perform consistently well in a wide variety of tasks. Such is the case of Collaborative Innovation Networks (COINs) defined as the core of collaborative knowledge. They are highly interdisciplinary collaborative networks, which combine a large number of fields such as CI or crowdsourcing [21], driven by swarm creativity, where people work together in a structure that allows the creation and fluid exchange of new ideas [21]. The CoSpace, are spaces created as a collaborative work tool for the needs of industry, which focuses primarily on creating collaborative engineering spaces to plan, design and build new products for the automotive, space and construction industries [22].

Our capacity to generate better solutions depends on teams of scientists, engineers, or knowledge workers and their networks [17], in fact, the interesting thing about these collaborative networks, no matter what they are called or where they are applied, is the group intelligence and the exchange of information and ideas. Within collaborative groups, it has been determined that there is a “general intelligence” that arises from correlations between people’s performance on a wide variety of cognitive tasks [23], similarly, it has been shown that the structure of social interactions can enhance or hinder the achievement of objectives [17]. In fact, the ability to solve complex problems could be greatly enhanced by improving the instrumental and expressive links between individuals [17].

Within Collective Intelligence, it has been found that individuals who participate in groups benefit. In fact it has been found that groups with high CI develop greater shared attention, transactive memory and a better problem-solving process [24, 25]. Thus, group learning emerges from collaborative work [26]. Collaborative teaching or also called co-teaching has been shown to be a pedagogical strategy of high value and potential as a support tool in the classroom, or as a strategy for the professional improvement of the individual’s skills and knowledge [27]. Edmondson et al. in [28] classifies team learning into three distinct foci: performance improvement, or the speed at which groups improve their performance; task mastery, or how team members coordinate knowledge and skills to accomplish tasks; and group process, or what drives learning-oriented behaviours and processes in teams.

In [29] it has been shown that groups with higher CI improve their performance rapidly, indicating that they have learned faster than groups with low CI. It is also suggested that even a moderate level of cognitive diversity helps to improve overall group performance for different tasks, as high levels of cognitive diversity hinders the transfer of information between members, thus hindering coordination and collaboration per se [29].

4 How Collective Intelligence Could Be Used to Gear Agile and Sustainability

In terms of Melo and Eckstein, continuous collaboration allows the self-organization of teams and therefore the discovery of new market needs [4] and as we have already seen in the previous sections, market needs have already changed and CI can be the gear that helps the transfer of knowledge through team learning to Agile teams. In other words, CI allows the introduction of a new Stakeholder which is the welfare of all citizens and can promote the adoption of new global objectives within Agile Methodologies through Team Learning, making Sustainability a new tacit or implicit requirement within any IT or Software development. The Table 1 shows how Agile principles and values fit with CI.

Table 1. Relation between Agile values/principles and collective intelligence

Agile value/principle	Relation with collective intelligence
Transparency	CI takes knowledge from the local context to the global context
Self-organization	CI promotes self-organized teams
Continuous learning	CI contributes to team learning
Constant customer focus	CI connects all the required stakeholders

Having new stakeholders and a new implicit requirement, it is possible to see that CI can work not only as a tool, but also as another member of the team, collaborating as Scrum Master/Coach regarding Team Learning, Organization and Values and as Product Owner transferring and curing the knowledge of the new stakeholders, to transfer them to the Agile Team. In this way the knowledge that is generated can be transferred from the local to the global context, adapting those solutions and therefore the Agile methodologies that work in the different contexts with the goal of achieving the global objectives of sustainability in the economic, environmental, and social axes, always keeping in mind the terms of use and sustainable development (Fig. 1).

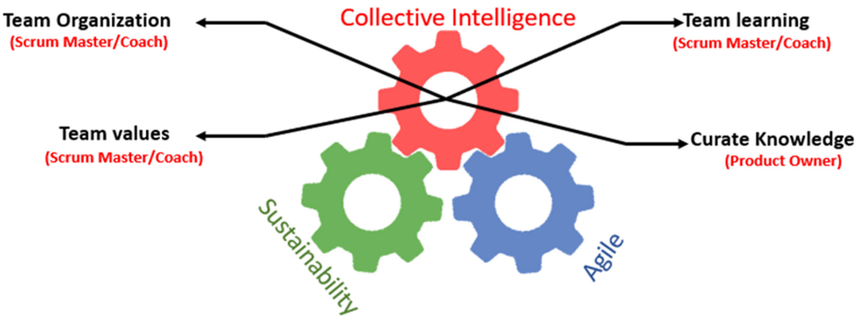


Fig. 1. CI gearing Agility with Sustainability

Usually teams with better performance or with higher CI generate better solutions and, in fact, CI is a good predictor of the performance that the group will have [29], Therefore, assessing or measuring CI can be an indicator that Agile teams are adopting sustainability goals.

Experiments can be made with Agile Teams to confirm that CI can effectively contribute to Agile teams, the results of the experiments can be measured from several dimensions: first, using the Team Diagnostic Survey (TDS), the collaborative work of Agile teams can be evaluated, to determine how they work, how their workflow is and how their internal processes can be optimized. Second, the TDS also allows to measure the interpersonal process and collaborative learning, so it would be possible to evaluate the increase of skills and the quality of learning of new sustainability concepts, as well as how these new concepts are transferred to the software during the development and deployment stages. This whole process can help to understand how to extend Agile methodologies so that they can be adjusted to the new sustainability goals. In the following section how, CI can be measured is described more in depth.

5 Measuring Collective Intelligence

To evaluate Collective Intelligence in terms of group performance, Woolley et al. in [23] applies a set of tasks extracted from the quadrants of "McGrath task Circumplex" [30], which is an established taxonomy for group tasks; such tasks may include visual puzzle solving, brainstorming, moral judgments and negotiating under limited resources. Engel et al. in [24], uses the task battery used by Woolley et al. in [23] and also the MacGrath [30] and Larson tasks [31]. In [25], psychological sensing is used to understand the collaboration dynamics. This sensing provides a finer degree of understanding about the participants' experience during the collaboration process, facilitating awareness among peers or partners [25]. Variables such as group satisfaction and cohesion have been considered as reliable indicators of the team's level of rapport, even in online collaborations. Furthermore, authors in [25], also propose measuring the group satisfaction, in order to determine if there is some correlation between CI and the satisfaction perceived by the members of the group, to achieve this, six items which reflects the quality of the group collaboration through an adaptation of the "Team Diagnostic Survey" [32] are used.

The Team Diagnostic Survey or TDS, is a tool used for assessing the properties of a team, and was specifically designed to be useful in scholarly research on teams and in the practical diagnosis of teams' strengths and weaknesses [32]. According to [32], the team effectiveness is must accomplish these criteria's:

- The productive output of the team meets or exceeds the standards of quantity, quality, and timeliness of the team's clients.
- The social processes the team uses in carrying out the work enhance members' capability to work together interdependently in the future.
- The group experience contributes positively to the learning and well-being of individual team members, rather than frustrating, alienating, or de-skilling them.

6 Conclusions and Future Work

Achieving sustainable goals, requires the highly collaborative effort and work of Agile teams and stakeholders, to adjust to the new market's needs, at this point Collective intelligence has demonstrated its potential in many areas when generating solutions to complex problems and extending local knowledge to a global scale, allowing the rapid adaptation of processes and methodologies. Collective intelligence, sustainability and Agility have a common point which is the transfer of knowledge, which can be measured through team learning.

Agile and sustainability share a common ground: they can supplement each other by an appropriate team collaboration and learning scheme. Collective intelligence can be used as a framework to convey this transformation. Collective intelligence, besides, can be of help~~,~~ to understand how teams work, and how work processes can be changed to get improved. All this, thanks to the use of tools such as TDS, that allow us to assess team performance.

CI has proven to fit into the Agile manifesto and its values and principles, so CI concepts can be applied and transferred to Agile methodologies to achieve the new global goals concerning sustainability. CI allows to introduce Sustainability within Agile methodologies as a new implicit requirement in software development, where the new stakeholders are the whole humanity welfare.

As mentioned in the previous sections, sustainability in Agile teams has a lot to do with group learning, where collective intelligence has proven to have tools that allow to enhance group learning and not only evaluate group work, but also provide feedback on those aspects that can be improved. All this in a framework of Collective Intelligence among Agile teams can allow the adoption of sustainability goals, within Agile methodologies, for subsequent adoption in a global context.

Finally, experiments are needed to confirm the potential of CI to engage the concept of Agile in Sustainability and, perhaps, to help update the Manifesto.

References

1. Transforming our world: the 2030 Agenda for Sustainable Development—Department of Economic and Social Affairs. https://sdgs.un.org/2030agenda
2. Melo, C.: Another purpose for agility: sustainability. In: Meirelles, P., Nelson, M.A., Rocha, C. (eds.) WBMA 2019. CCIS, vol. 1106, pp. 3–7. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-36701-5_1
3. Manifesto for Agile Software Development. https://Agilemanifesto.org/
4. Eckstein, J., Melo, C.O.: Sustainability: delivering agility's promise. In: Calero, C., et al. (eds.) Software Sustainability. Springer (2021, Submitted)
5. Felipe, C.M., Roldán, J.L., Leal-Rodríguez, A.L.: Impact of organizational culture values on organizational agility. Sustain. **9**, 2354 (2017). https://doi.org/10.3390/su9122354
6. Obradovićl, V., Todorović, M., Bushuyev, S.: Sustainability and agility in project management: contradictory or complementary? In: 2018 IEEE 13th International Scientific and Technical Conference on Computer Sciences and Information Technologies, CSIT 2018 – Proceedings, pp. 160–164 (2018). https://doi.org/10.1109/STC-CSIT.2018.8526666
7. Sommer, A.F.: Agile Transformation at LEGO Group: implementing Agile methods in multiple departments changed not only processes but also employees' behavior and mindset. Res. Technol. Manag. **62**, 20–29 (2019). https://doi.org/10.1080/08956308.2019.1638486
8. Hemon, A., Lyonnet, B., Rowe, F., Fitzgerald, B.: From agile to DevOps: smart skills and collaborations. Inf. Syst. Front. **22**(4), 927–945 (2019). https://doi.org/10.1007/s10796-019-09905-1

9. Malone, T.W., Bernstein, M.: Handbook of Collective Intelligence. MIT Press, Cambridge (2015)
10. Diegmann, P., Rosenkranz, C.: Team performance in agile software development projects: the effects of requirements changes, time pressure, team diversity, and conflict. Int. Res. Work. IT Proj. Manag., 2 (2017). https://aisel.aisnet.org/irwitpm2017/2
11. Hilty, L.M., Aebischer, B.: ICT for sustainability: an emerging research field. In: Hilty, L.M., Aebischer, B. (eds.) ICT Innovations for Sustainability. AISC, vol. 310, pp. 3–36. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-09228-7_1
12. Letouzé, E., Pentland, A.: Towards a human articial intelligence for human development (2019)
13. Elia, G., Margherita, A., Passiante, G.: Digital entrepreneurship ecosystem: how digital technologies and collective intelligence are reshaping the entrepreneurial process. Technol. Forecast. Soc. Change **150** (2020). https://doi.org/10.1016/j.techfore.2019.119791
14. Lameras, P., Petridis, P., Dunwell, I.: Raising awareness on sustainability issues through a mobile game. In: Proceedings of 2014 International Conference on Interactive Mobile Communication Technologies and Learning, IMCL 2014, pp. 217–221 (2015). https://doi.org/10.1109/IMCTL.2014.7011135
15. Bonci, A., Clini, P., Martin, R., Pirani, M., Quattrini, R., Raikov, A.: Collaborative intelligence cyber-physical system for the valorization and re-use of cultural heritage. J. Inf. Technol. Constr. **23**, 305–323 (2018)
16. Peeters, M.M.M., et al.: Hybrid collective intelligence in a human–AI society. AI Soc. **36**(1), 217–238 (2020). https://doi.org/10.1007/s00146-020-01005-y
17. De Montjoye, Y.-A., Stopczynski, A., Shmueli, E., Pentland, A., Lehmann, S.: The strength of the strongest ties in collaborative problem solving (2014). https://doi.org/10.1038/srep05277
18. Malone, T.W.: Superminds: The Surprising Power of People and Computers Thinking Together. Little, Brown, Boston (2018)
19. Fan, W., Wang, W., Xiao, T.: Multidisciplinary collaboration simulation optimization platform for complex product design. In: 2007 2nd International Conference on Pervasive Computing and Applications, ICPCA 2007, pp. 174–178 (2007). https://doi.org/10.1109/ICPCA.2007.4365434
20. Barlow, J.B., Dennis, A.: Not as smart as we think: a study of collective intelligence in virtual groups (2014)
21. Gloor, P.A., Riopelle, K., Gluesing, J., Lassenius, C., Paasivaara, M., Garcia, C.: Int. J. Organ. Des. Eng. **2**, 127–131 (2012)
22. Patel, H., Pettitt, M., Wilson, J.R.: Factors of collaborative working: a framework for a collaboration model (2012). https://doi.org/10.1016/j.apergo.2011.04.009
23. Woolley, A.W., Chabris, C.F., Pentland, A., Hashmi, N., Malone, T.W.: Evidence for a collective intelligence factor in the performance of human groups. Science **80**(330), 683–686 (2010). https://doi.org/10.1126/science.1193147
24. Engel, D., et al.: Collective intelligence in computer-mediated collaboration emerges in different contexts and cultures. In: Conference on Human Factors in Computing Systems – Proceedings 2015-April, pp. 3769–3778 (2015). https://doi.org/10.1145/2702123.2702259
25. Chikersal, P., Tomprou, M., Kim, Y.J., Woolley, A.W., Dabbish, L.: Deep structures of collaboration: physiological correlates of collective intelligence and group satisfaction. In: Proceedings of ACM Conference on Computer Supported Cooperative Work and Social Computing, CSCW, pp. 873–888 (2017). https://doi.org/10.1145/2998181.2998250
26. Giacomell, G.: Augmented collective intelligence: human-AI networks in a virtual future of work (2020)

27. Cotrina García, M., García García, M., Caparrós Martín, E.: Ser dos en el aula: las parejas pedagógicas como estrategia de co-enseñanza inclusiva en una experiencia de formación inicial del profesorado de secundaria. Aula Abierta **46**, 57 (2017). https://doi.org/10.17811/rifie.46.2017.57-64
28. Edmondson, A., Dillon, J., Roloff, K.: Three perspectives on team learning: outcome improvement, task mastery, and group process. Acad. Manag. Ann. **1**, 269–314 (2007). https://doi.org/10.1080/078559811
29. Woolley, A.W., Aggarwal, I., Woolley, A.W., Aggarwal, I.: Collective intelligence and group learning. In: Oxford Handbook of Group and Organizational Learning, pp. 490–504 (2020). https://doi.org/10.1093/oxfordhb/9780190263362.013.46
30. McGrath, J.: Groups: Interaction and Performance. Prentice-Hall, Englewood Cliffs (1984)
31. Larson Jr., J.R.: In Search of Synergy in Small Group Performance. Psychology Press, New York (2010)
32. Wageman, R., Hackman, J.R., Lehman, E.: Team diagnostic survey: development of an instrument. J. Appl. Behav. Sci. **41**, 373–398 (2005). https://doi.org/10.1177/0021886305281984

Summary of 4th International Workshop on Software-intensive Business

Fueling a Software-driven Economy: The 4th International Workshop on Software-intensive Business

Karl Werder[1], Sami Hyrynsalmi[2], and Xiaofeng Wang[3]

[1] University of Cologne, 50969, Cologne, Germany
werder@wiso.uni-koeln.de
[2] Lappeenranta University of Technology, 53850, Lappeenranta, Finland
Sami.Hyrynsalmi@lut.fi
[3] Free University of Bozen-Bolzano, 39100, Bolzano, BZ, Italy
xiaofeng.wang@unibz.it

Abstract. The global pandemic has shown: thanks to advanced software technologies, society and businesses were able to quickly respond to environmental disruptions. Software-intensive businesses had to quickly pivot their business models, and demands in software-based service offerings facilitating remote work drastically increased and challenged control modes of prior management practices. These example challenges cannot be tackled by engineering or business discipline alone. The International Workshop on Software-intensive Business (IWSiB) brought together different research communities working on topics relevant to software-intensive businesses to jointly investigate these challenges. For example, the keynote speaker Professor Dr. Guenther Ruhe took a return-on-investment perspective on machine learning. The workshop facilitated knowledge exchange through discussions related to software platforms, software startups, software pricing, and the dark web. The feedback from the wider community helped the authors of the eight papers presented at the workshop to further develop and improve their research.

Keywords: Software-intensive business · Software platform · Software startup

1 Introduction

The global pandemic has shown, thanks to advanced software technologies, society and businesses were able to quickly respond to environmental disruptions. Demands in software-based service offerings facilitating remote work drastically increased. For example, the revenue for Check Point, a company specializing on VPN solutions, increased by 4% in 2020 [1]. The revenue of Zoom, a video conferencing solution, even jumped by 30% in 2020 [2]. The literature has widely acknowledged the positive effects of an organization's ability to use software technology in order to quickly react to changes in its environment [3]. Consequently, software producing organizations face increasing demands and evolving requirements from businesses that seek innovative solutions for new realities that are enforced by physical distancing guidelines.

Society and businesses have learned they are less dependent on physical presence. Rather, they experienced that software helped them in many novel ways that they had not explored before. For example, software startups have shown their ability to quickly pivot, thanks to the malleability of their product's [4]. The german startup "rausgegangen" was a software-based startup to find venues and events in one's area. Needless to say that this startup was particularly hit by the COVID19 restrictions, so the company pivoted to "dringeblieben" a platform that let people connect to artists and entertainers, so they can join their events through an online stream. This is only one of many examples we all experienced in the last year, showing that software is increasingly central to ever more business endeavours. Hence, this year's International Workshop on Software-intensive Business seeks to advance knowledge by "Fueling a Software-driven Economy".

We also experienced how, many businesses "went online" and quickly sent their employees to work from home. Many learned the upsides and downsides of this. Some enjoyed the increased autonomy, others felt more pressured and stressed. In addition, managers had to find new modes of control in order to check development progress and project status. This was particularly challenging, as co-location is often associated with positive performance implications for agile software development [5]. Both examples show how software-intensive businesses have been affected by environmental changes. Thus, they had to identify and develop adequate responses that assured their organization's survival.

Overall, the software-intensive business community seeks to better understand and explain these challenges, as they affect different aspects of software-intensive businesses: managing the technology, the people, and their interaction. First, software-intensive businesses need to manage the technology. Nowadays, many software-intensive businesses offer entire platforms that are increasingly complex. Architectural and technological questions are important, as they have important cost and quality implications. Second, people are involved and needed in the development and design of software. Management challenges include organizational aspects, as development organizations quickly consist of thousands of employees with many more being involved in their ecosystem. Also, team and individual questions are important, as development organizations often consist of multiple teams that need to communicate and develop a shared mindset, but also individual needs are important in order to guarantee well being and job satisfaction. Third, the relationship between organizational and technological aspects provide many challenges. For example during adoption, users need to develop trust toward the systems, but also during the development, where organizational and technical structure relate to each other.

Hence, the scientific community of software-intensive business researchers investigate "sustainable software-based value creation, capture, and delivery through arrangements and methods i) within organizations (e.g., product management, business models, agility) and ii) between organizations (e.g. ecosystems, platforms, app stores, OSS communities)" [6]. Given the unique characteristics of software and the important implications it has toward business, The workshop brought together a heterogenous group of researchers from different research sub-fields such as software engineering economics, software product management, software ecosystems, technology management, software platforms, or software startups. This heterogeneity is the strength of this community, as its challenges stem from prior work in fields.

The 4th *International Workshop on Software-intensive Business* (IWSiB) was co-organized with XP 2021 with participants from the US and all over Europe. The workshop had around 30 participants that joined in discussions on software platforms, software startups, software pricing and the dark web. As such, it provided a venue for members of the software engineering and business research communities to discuss issues, exchange information and experiences. The workshop also encouraged participants to share early work and work-in-progress to obtain feedback from the wider community and strengthen the results and contributions of their research projects. Additionally, the workshop provided a platform for researchers and practitioners to meet and develop new project ideas.

2 The State of Software-intensive Business Research

This year's workshop featured a keynote, followed by three sessions on software platforms, software startups, and software pricing. Prof. Dr. Guenther Ruhe from the University of Calgary gave the keynote on a return-on-investment perspective on machine learning in software-intensive business. The talk highlighted the widespread use of machine learning in software engineering and its potential to improve efficiency and effectiveness. Yet, one challenge of machine learning is its need to make proper adjustments to the solution approach. Prof. Ruhe suggested the need to take a financial perspective on these efforts. He already applied a return-on-investment perspective to prior instantiations of machine learning in the context of software requirements dependency extractions. In particular, an open-source software dataset was analyzed for decisions about dependencies. Based on the solutions accuracy and return-on-investment analysis, Prof. Ruhe proposed recommendations for scholars that seek to benefit from machine learning in the future.

Thereafter, the workshop facilitated discussions along three sessions. First, three studies related to software platform research were discussed. Jaakko Vuolasto and Kari Smolander presented their study on "Genesis of a Wood Harvesting B2B Software Platform". They conducted a qualitative study on the use of digital platforms in business-to-business relationships in the context of Finland's forestry industry. They identified three insights: power shifts of different stakeholders as the platform evolved, governance mechanisms need to be contextualized, and the evolution of the platform needs to account for different roles of complementors in the ecosystem. Robert Evertse, Abel Lencz, Tea Sinik, Slinger Jansen, and Lamia Soussi presented their work on "Is your Software Ecosystem in Danger? Preventing Ecosystem Death through Lessons in Ecosystem Health". The study discusses four popular software ecosystem cases of the mobile phone industry. As a result, the authors derive 7 demise principles with corresponding countermeasures. For example, competitors are often underestimated and to counter this demise, organizations should increase their competitive advantage through increased market awareness. Virginia Springer and Dimitri Petrik presented their work on “A Taxonomy of Price Parameters for Digital Platforms”. The authors conducted a literature review in order to develop a new taxonomy. The taxonomy consists of five dimensions, 13 parameters with 2–4 characteristics each. For example, the dimension revenue model consists of the parameters pricing model, subsidization and pie-splitting. Following this session of software platforms, the workshop had a short break before continuing.

Second, three studies related to software start-up research were discussed. Jorge Melegati presented his work on “Towards a framework to guide the development of practices for software startups”. The study reviews the literature on success of software projects and success of startups to propose a framework that guides the creation of practices for software startups. The use of the framework is demonstrated using different examples. Tor Sporsem, Anastasiia Tkalich, Nils Brede Moe, Marius Mikalsen and Nina Rygh presented their work on “Using Guilds to Foster Internal Startups in Large Organizations: A case study”. The authors conducted a single case study to investigate the use of guilds in internal startups. The results identified three software product innovation challenges and achievements of the guild. For example, when the startup lacks knowledge on building and scaling products, the guild can help improve the coordination with the software development unit. In a similar vein, Anastasiia Tkalich, Nils Brede Moe and Tor Sporsem presented their work on “Employee-Driven Innovation to Fuel Internal Software Startups”. The study investigated two cases through interviews, meetings notes and other documents to understand what characterizes employee-driven innovation in internal software startups. The results suggest seven contextual characteristics, such as a dedicated product manager and innovation coaching, that foster employee-driven innovation.

Third, two studies related to pricing practices and the dark web were discussed. Andrey Saltan and Kari Smolander presented their work on “SaaS Pricing Practices Typology: A Case Study”. The multi-case study investigates SaaS pricing practices to propose a new typology. The resulting typology consists of four factors: Targeted types of customers and market segments, Perceived value and WTP, The complexity of SaaS purchase and usage and Level of SaaS nicheness. Samuel Onyango, Emilie Steenvoorden, Joram Scholten, and Slinger Jansen presented their work on “Assessing the Health of the Dark Web: An Analysis of Dark Web Open Source Software Projects”. The authors derived health assessment criteria and applied these to two open-source communities that are associated with the dark web, i.e., tor and i2p. The results suggest that both communities have seen an increase in user activities in recent years, suggesting a positive health assessment. Following the third session, we had a closing session that asked the participants for their feedback and potential future improvements of the workshop.

3 Future Challenges for Software-intensive Business Research

While the workshop itself focused on topics related to software platforms, software startups, and software pricing, we also discussed potential future challenges. Based on these discussions and our own experience, we propose three challenges that require more efforts by the software-intensive business research community:

- **The AI workforce**—Software-intensive business scholars focused on value-creation, capture and delivery within the organization [6]. For example, prior studies on organizational management in software development investigated emotions [7, 8], agility [9, 10], or organizational failures [11]. We suggest that these behaviorally anchored research topics need to be revisited in the uprising of Artificial Intelligence (AI). For example, AI shapes the way in which organizations develop software, as bots, for example, assist in coordinating software projects (e.g., [12]). The interaction between developers and AI can range from reflexive or supervisory behavior with limited decision making authority up to anticipatory or prescriptive behavior in which the AI receives increasing autonomy within decision-making processes [13]. This research area is still in a nascent stage and requires more attention from software-intensive business scholars. Scholars can ask, for example, how can people and AI collaborate in new business processes to deliver value?
- **Digital collectives**—Digital collectives, that is, a cooperative enterprise facilitated by digital technologies, received increasing attention in the past. Within the software-intensive business community, scholars investigated open innovation communities [14] and open source software communities [15]. More recently, organizations were forced to send their employees into quarantine, having their workforce operate from their homes. As vaccination rates increase and the spread of the pandemic slows down, we already start seeing corporations asking their employees to return to their offices. However, as the pandemic has so vividly shown us, employees expect more freedom in working from home and we suggest that digital collectives become increasingly important for organizations in their day-to-day operations. As such, scholars could investigate how organizations and workers find new configurations in digital collectives for their daily operations?
- **Platform innovation**—platforms have been a central research topic of the software-intensive business community. Scholars investigated their architecture [16], health [17], and management [18]. More recently, also ethical concerns have been taking into consideration [19]. However, little research is available on the evolution of these platforms. For example, how do we evolve platforms for continuous innovation? Particularly competitive pressures and the diminishing value of software that remains stable demand for continuous innovation of platforms.

References

1. Cohen, T.: Check Point profit tops estimates but uncertainty clouds outlook. https://www.reuters.com/article/us-chk-pnt-sftwre-results/check-point-profit-tops-estimates-but-uncertainty-clouds-outlook-idUSKCN24N1WV. Accessed 25 Jan 2021
2. Malara, N., Nellis, S.: Zoom forecasts sales surge as video conferencing becomes a daily routine. https://www.reuters.com/article/zoom-video-commn-results-int-idUSKBN25R2QV. Accessed 25 Jan 2021
3. Werder, K., Richter, J., Hennel, P., Dreesen, T., Fischer, M., Weingarth, J.: A three-pronged view on organizational agility. IT Prof. **23**, 89–95 (2021). https://doi.org/10.1109/MITP.2020.3016488
4. Teutenberg, J.: How a new venture identified digital opportunities in the COVID-19 Crisis to transform their business model. In: Hovestadt, C., Recker, J., Richter, J., Werder, K. (eds.) Digital Responses to Covid-19, pp. 105–117. Springer International Publishing, Heidelberg (2021). https://doi.org/10.1007/978-3-030-66611-8_8
5. Hummel, M., Rosenkranz, C., Holten, R.: The role of communication in agile systems development: an analysis of the state of the Art. Bus. Inf. Syst. Eng. **5**, 343–355 (2012). https://doi.org/10.1007/s12599-013-0282-4
6. Abrahamsson, P., Bosch, J., Brinkkemper, S., Mädche, A.: Software Business, Platforms, and Ecosystems: Fundamentals of Software Production Research. Dagstuhl, Germany (2018). https://doi.org/10.4230/DagRep.8.4.164
7. Graziotin, D., Wang, X., Abrahamsson, P.: Happy software developers solve problems better: psychological measurements in empirical software engineering. PeerJ. **2**, e289 (2014). https://doi.org/10.7717/peerj.289
8. Werder, K.: The evolution of emotional displays in open source software development teams: an individual growth curve analysis. In: 2018 IEEE/ACM 3rd International Workshop on Emotion Awareness in Software Engineering (SEmotion), pp. 1–6. ACM Press, Gothenburg, SE (2018). https://doi.org/10.1145/3194932.3194934
9. Vidgen, R., Wang, X.: Coevolving systems and the organization of agile software development. Inf. Syst. Res. **20**, 355–376 (2009). https://doi.org/10.1287/isre.1090.0237
10. Werder, K., Maedche, A.: Explaining the emergence of team agility: a complex adaptive systems perspective. Inf. Technol. People. **31**, 819–844 (2018). https://doi.org/10.1108/ITP-04-2017-0125
11. Giardino, C., Wang, X., Abrahamsson, P.: Why early-stage software startups fail: a behavioral framework. In: Lecture Notes in Business Information Processing, pp. 27–41 (2014). https://doi.org/10.1007/978-3-319-08738-2_3
12. Hukal, P., Berente, N., Germonprez, M., Schecter, A.: Bots coordinating work in open source software projects. Computer (Long. Beach. Calif). **52**, 52–60 (2019). https://doi.org/10.1109/MC.2018.2885970

13. Baird, A., Maruping, L.M.: The next generation of research on is use: a theoretical framework of delegation to and from agentic is artifacts. Mis Q. **45**, 315–341 (2021). https://doi.org/10.25300/MISQ/2021/15882
14. Munir, H., Wnuk, K., Runeson, P.: Open innovation in software engineering: a systematic mapping study. Empir. Softw. Eng. **21**, 684–723 (2016). https://doi.org/10.1007/s10664-015-9380-x
15. Lindberg, A., Berente, N., Gaskin, J., Lyytinen, K.: Coordinating interdependencies in online communities: a study of an open source software project. Inf. Syst. Res. **27**, 751–772 (2016). https://doi.org/10.1287/isre.2016.0673
16. Anvaari, M., Jansen, S.: Architectural openness: comparing five mobile platform architectures. In: Software Ecosystems, pp. 138–158. Edward Elgar Publishing. https://doi.org/10.4337/9781781955635.00016
17. Jansen, S.: Measuring the health of open source software ecosystems: beyond the scope of project health. Inf. Softw. Technol. **56**, 1508–1519 (2014). https://doi.org/10.1016/j.infsof.2014.04.006
18. Foerderer, J.: Interfirm exchange and innovation in platform ecosystems: evidence from apple’s worldwide developers conference. Manage. Sci. **66**, 4772–4787 (2020). https://doi.org/10.1287/mnsc.2019.3425
19. Hyrynsalmi, S., Koskinen, J.S.S., Hyrynsalmi, S.M.: A review of ethical discussions on platforms and ecosystems. In: CEUR Workshop Proceedings, pp. 9–19 (2019)

SaaS Pricing Practices Typology: A Case Study

Andrey Saltan[1,2](✉) and Kari Smolander[1]

[1] LUT University, Lappeenranta, Finland
{andrey.saltan,kari.smolander}@lut.fi
[2] HSE University, St. Petersburg, Russia

Abstract. Software-as-a-Service (SaaS) pricing addresses decisions of monetary compensation and the conditions for the SaaS solution to the customer. Efficient SaaS pricing requires sophisticated decision-making and analytics, as well as coordination and compromises between the many business functions involved. The decision-making includes integrated analysis of different perspectives and streams of information. Like in many other product management areas, there is no silver-bullet solution for pricing. We conducted a multiple case study using fifteen SaaS companies with data collection primarily through semi-structured interviews to assess SaaS pricing practices and identify major factors that affect the way pricing is done. We identified four distinct types of SaaS pricing patterns and detailed their main characteristics.

Keywords: Software-as-a-Service · Decision-making · Pricing · Case study

1 Introduction

Software-as-a-Service (SaaS) pricing refers to the entire scope of decisions, practices, underlying conditions, and processes that determine the monetary compensation for using SaaS solutions. It is an essential and challenging element of SaaS product management, with a significant impact on business success. Incorrect pricing can lead to market failure, even for a technologically advanced SaaS solution. Pricing serves as an essential bridge between different business functions (e.g., product planning and development, revenue and cost management, and customer acquisition and retention) and business units (e.g., R&D, product management, sales, and marketing). Recent studies and reviews indicate the progress and sophistication in SaaS pricing and the growing attention from practitioners. Multiple challenges for companies can still be identified that require support from the research community [1].

Overwhelming and complex pricing-related processes and structures, the unclear segregation of responsibilities for pricing between managers involved, premature decision-making practices, and constantly changing objectives are often prime challenges. Efficient pricing requires developing sophisticated multi-layered structures with many different mechanisms and options, considering the trade-offs, objectives, and outcomes that pricing must meet. Informed SaaS pricing decision-making requires the involvement of different stakeholders and the consideration of many factors that include market characteristics, product and technology specifications, customers, and customer

P. Gregory and P. Kruchten (Eds.): XP 2021 Workshops, LNBIP 426, pp. 87–95, 2021.
https://doi.org/10.1007/978-3-030-88583-0_8

needs and expectations. Taking into account these factors requires collecting a vast amount of data and advanced analysis, tasks that are not trivial.

Existing publications by scholars and practitioners reveal the variety and complexity of mechanisms available for SaaS companies while pricing their solutions [2–5]. They also provide an overwhelming number of recommendations concerning different pricing aspects [1]. However, the repeated enumeration of possible pricing options and fragmented recommendations does not bring the required clarity to SaaS companies, and pricing remains one of the most under-managed functions in many of them. Little evidence exists about the interconnection of different components of SaaS pricing, typologies of overall pricing practices, or decision-making organization principles.

This paper aims to identify and evaluate patterns in SaaS pricing, identify the major factors that affect it, and propose a typology of SaaS pricing practices. This study continues our inquiry into how SaaS companies design and deploy their pricing practices and processes.

2 Background

2.1 Related Studies

SaaS pricing is a maturing and prominent area of research. Existing SaaS pricing studies indicate the progress and sophistication in SaaS pricing practices and offer solutions that can carry SaaS pricing state-of-the-practice to a higher level. Our recent multivocal literature review [1] identified multiple challenges that require further support from the research community.

Some studies have already adopted the case-study method to evaluate various pricing aspects in SaaS and software companies. For example, based on interviews with software professionals from multiple case companies, Ojala [6, 7] identified and assessed factors that affect selecting revenue and pricing models in software companies. In another study [8], Ojala and Laatikainen investigated the interrelation between SaaS architecture and SaaS pricing practices.

2.2 SaaS Pricing

Existing pricing state-of-the-art and state-of-the-practice suggest distinguishing between four main pricing strategies: value-based pricing, market-based pricing, competitor-based pricing, and cost-based pricing. In short, they can be explained as follows in the SaaS context. Value-based pricing assumes aligning prices with the value perceived by the customer. Market-based pricing is grounded in an analysis of the market equilibrium of all customers and SaaS providers. Competitor-based pricing assumes aligning prices with the prices offered by competitors with the premium or discount depending on the circumstances. Finally, Cost-based Pricing suggests setting prices based on the cost structure of SaaS providers. In application to SaaS, researchers and practitioners have repeatedly emphasized the advantages and importance of value-based pricing. However, all four pricing strategies might exist in practice, and in many cases, the actual strategy is a hybrid combination of these strategies.

Several frameworks and structures exist to organize and systematize pricing in application to SaaS and cloud solutions in general [1]. However, in our study, we adopted a more generic, widely accepted, and comprehensive one called the Strategic pricing pyramid [9, 10]. The framework has the following levels from the bottom up:

Value Creation: The logic of value generation for customers from using the SaaS solution, including the metrics of impact of specific parameters on value.
Price Structure: The logic of structuring prices for a given SaaS solution, including principles of price variability depending on the customer-specific parameters.
Price and Value Communication: The principles of price and value communication to customers.
Pricing Policy: The principles of how prices may be altered, by whom, under what circumstances, and to what degree.
Price Level: The actual charge within the price structure according to the pricing policy.

3 Research Method

The following research question drove our study: *What types of SaaS pricing practices can be identified in a real-life context?* To address this question, we used a multiple case study research design to compare existing SaaS pricing practices and processes [11]. The case sampling strategy was guided by the diverse case approach with its primary objective to achieve variance along the relevant dimensions. Our scope of companies includes two major types of SaaS providers, "born-in-the-cloud" companies that usually have just one flagship SaaS solution and large IT vendors or traditional enterprise software vendors looking to expand into SaaS software markets. Other dimensions, including company size and maturity, target market type, maturity, and location, were considered while selecting case companies.

We selected a set of fifteen primary and secondary cases. Our primary cases include companies whose pricing managers we interviewed. Most of them are "born-in-the-cloud" small and medium-sized companies that usually have just one flagship SaaS solution. We could not involve large US-based SaaS companies in our study, although their presence is essential to understand and develop a comprehensive SaaS pricing typology.

To remedy this situation, we decided to include cases that we did not interact directly with. We assessed their pricing practices through available information and teaching cases on their business strategies and operations. We referred to these cases as secondary and found them in the Case Center[1], the largest repository of teaching cases. This allowed us also to make assessments of pricing in large SaaS and digital companies as well as in enterprise software vendors with SaaS solutions in their product portfolio. An overview of the primary and secondary case companies is summarized in Table 1.

The goal is to identify decision-making practices and processes and understand the logic behind them. A within-case analysis was conducted with the analytical strategy of explanation-building based on case descriptions. The case analysis can be classified

[1] https://www.thecasecentre.org/.

as exploratory. We developed patterns and categories and identified similarities and differences in the data. The logical sequence followed the research goals, starting with within-case analysis to establish themes and then continued by a cross-case comparison to identify similarities and differences.

Table 1. Characteristics of case companies

Case	Case type	Number of employees	Number of SaaS solutions	Market type
A	Primary	<10	1	B2B
B	Primary	<10	1	B2B
C	Primary	<10	1	B2B & B2C
D	Primary	11–50	1	B2B & B2C
E	Primary	11–50	1	B2B & B2C
F	Primary	11–50	1	B2B
G	Primary	11–50	1	B2B
H	Primary	51–200	2	B2B
I	Primary	51–200	1	B2B
J	Primary	51–200	2	B2B
K	Primary	201–500	2	B2B
L	Primary	201–500	3	B2B
M	Secondary	51–200	1	B2B
N	Secondary	1001–5000	5	B2B & B2C
O	Secondary	201–500	3	B2C

For primary cases, the data collection consisted of interviews with SaaS managers responsible for pricing. The length of interviews varied from 1 to 2 h. The goal of the interviews was to identify the pressure points of decision-making in SaaS pricing, motivate companies to participate in the longitudinal study, and assess both the current status quo and product managers' perceptions of existing processes and practices. The data we obtained covered the following topics:

General information about the company and SaaS solution: name, industry, market, number of employees, number of customers, maturity level, business model, number of SaaS solutions, SaaS solution type, maturity level, etc.
SaaS pricing practices and processes: Pricing frameworks used, product activities allocation across business units, collaboration principles between business units, pricing tools used, SPM performance assessment principle, etc.
SaaS pricing decision-making principles: formal regulation and written policies on SaaS pricing activities, risks, and uncertainty identified, types of data collected for pricing decision-making, models and tools used to process provided data, information system support for pricing processes, etc.

For secondary cases, the data collection consisted of content analysis of the documented teaching cases and teaching notes to extract similar information.

4 A Typology of SaaS Pricing Practices

The qualitative research approach with semi-structured interviews allowed us to identify four major factors that affected SaaS pricing. The factors were the following:

Factor 1: types of customers and market segments targeted. We can distinguish between B2B, B2G, and B2C customers, as well as the size of targeted customers (especially in the B2B market).
Factor 2: delivered value and willingness to pay (WTP) for the SaaS solution. Specific estimates based on a limited number of cases are difficult to make; still, conventionally, we can distinguish between SaaS solutions with an average monthly usage fee of up to 100 USD, SaaS solutions with an average fee of more than 5000 USD, and those in between these two price levels.
Factor 3: the complexity of SaaS purchase and usage. We can distinguish between self-service SaaS solutions, SaaS solutions that might require human assistance in the purchase, customization, and maintenance, and SaaS solutions that require intensive human involvement, including offering additional professional and training services.
Factor 4: the level of nicheness of the SaaS solution. We can distinguish between mass-market SaaS solutions focused on solving problems typical for a wide range of customers and SaaS solutions focused on solving issues specific for customers from the same industry, country, or facing similar regulatory constraints.

Based on the analysis of these four factors, we developed a typology of four generic SaaS pricing approaches that we labeled *Mass-market SaaS pricing*, *Generalist SaaS*

Table 2. Typology of SaaS companies based on pricing practices

	Mass-market SaaS pricing	Generalist SaaS pricing	Specialist SaaS pricing	High-rise SaaS pricing
Case companies	**C, D, E, N, O**	**I, L, M**	**A, B, F, J, K**	**G, H**
F1: Targeted types of customers and market segments	B2C and B2B	B2B	B2B	Large B2B, B2G
F2: Perceived value and WTP	Low value and WTP	Low or moderate value and WTP	Moderate or high value and WTP	High value and WTP
F3: The complexity of SaaS purchase and usage	Self-service	Self-service	Moderate human involvement	High human involvement
F4: Level of SaaS nicheness	Mass-market	Mass-market	Niche-market	Niche Market

pricing, *Specialist SaaS pricing*, and *High-rise SaaS pricing*. While typology was based on our investigation of SaaS company pricing, it also appears reasonable from a general business model perspective as it represents different business models and pricing practices. These four pricing approaches are presented in Table 2 and described below.

Mass-market SaaS pricing refers to pricing practices often implemented in SaaS companies that offer mass-market solutions and operate in the B2C market and B2B market, focusing on small-sized companies. Such SaaS solutions might also be used in large companies as a part of private initiatives by small teams and individuals. The main pricing objectives for this type of pricing are customer acquisition, market share maximization, and winning the competition. A value-based pricing approach, to a large extent, is supplemented with market-based pricing. Companies of this type also often adopt the freemium model and a free model with monetization other than charging customers (i.e., advertisement). Adjusting for the level of company and SaaS solution maturity, the pricing-related processes can be highly formalized, driven by data analytics, and even automated.

Generalist SaaS pricing is often implemented in SaaS companies that offer mass-market services for customers on the B2B market, serving both small, mid-sized, and large companies. The main pricing objective for this type of pricing is customer acquisition, monetization and retention and winning the competition. Companies with this type of pricing employ a hybrid pricing approach based on a combination of value-based pricing and competitor-based pricing. While competing companies might evaluate and structure perceived value differently, the average amount of money charged per customer or account are quite similar. Instead of freemium in the case of mass-market SaaS pricing**,** companies with generalist SaaS pricing often use penetration pricing and sophisticated usage-based tiered pricing with multiple available options. Pricing-related processes are often formalized and driven by data analytics. Pricing automation may be employed; however, a sales team exists, and large companies can negotiate pricing individually.

Specialist SaaS pricing refers to pricing practices implemented by B2B SaaS companies that have a niche SaaS solution. The limited market requires more focusing on monetization and retention of existing customers with a high-quality service rather than acquiring new customers. Companies with this type of pricing implement value-based pricing in its canonical understanding with a fair match of prices to the value perceived. As a result, defining value metrics and assessing perceived value is crucial. However, most pricing-related processes are not usually formalized. Decision-making data can consist of direct feedback from customers. The basic pricing information might be publicly available; however, purchase processes typically involve interaction with the sales team.

High-rise SaaS pricing is implemented in companies aiming to serve large organizations with their SaaS solution. The main pricing objectives are customer monetization and retention along with sustainable business development. This type of SaaS pricing involves combining value-based pricing with cost-based pricing. The complexity of these SaaS solutions and the requirements for reliability and security means the associated costs might be quite high. Therefore, it is essential for companies with this type of pricing to ensure that revenue from a reasonably limited number of customers with high charges per account will cover these costs. Most of the pricing-related processes are not formalized, pricing contract terms are discussed individually with all customers, and the required supplementary services define the final price to a large extent. Pricing information is not publicly available.

The literature discusses and proposes many factors that should be considered while designing and implementing pricing. As part of the multivocal study, we revealed 24 factors and classified them into four categories: Market, Company, Consumers, Product [1]. However, the impact of these factors and the aspects of pricing they affect remained unclear. Factors 1–4 correspond with the most cited factors as specified in [1]. While Factors 1 and 2 have a direct match, Factors 3 and 4 can be considered subfactors of a broader factor "functions and features" in the Product category.

Besides these four factors, product/company maturity, cost structure, and type of solution might affect and explain pricing practices in SaaS companies. However, our qualitative analysis suggests that maturity and costs could explain pricing practices ex-post rather than define them ex-ante. These factors set certain constraints and limitations on companies and managers; however, various companies overcome these constraints and limitations differently. As for the type of the solution, it was not clear how this could be determined and generalized from the case study as we covered only several categories of SaaS solutions from the extensive hierarchy (i.e., G2 software category hierarchy[2]). As a result, we decided not to incorporate these three factors in the typology.

5 Discussion and Practical Implications

The results of our study contribute to the understanding of pricing practices. We aimed to answer the research question of what types of SaaS pricing practices can be identified in a real-life context. To answer this question, we adopted a case-study research approach to explore pricing in fifteen SaaS companies. As a result, we developed a taxonomy of pricing practices. This typology can serve as a foundation for designing and establishing pricing practices in SaaS companies.

Our findings suggest that major factors of pricing in SaaS companies are the following: the targeted types of customers and market segments, the perceived value and willingness to pay for the SaaS solution, the complexity of the SaaS solution and its adoption by customers, and the level of nicheness of the SaaS solution. While the typology was based on an assessment of SaaS pricing practices, it can also be interpreted from the perspective of SaaS companies' business models.

Several implications for SaaS companies can be derived from our study. Gaining a clear understanding of pricing complexity for a given SaaS business model is essential to its long-term viability. While certain types of SaaS pricing practices can be identified, there is still no silver bullet. Within each recognized type, practices may vary depending on many different factors (i.e., product/company maturity) and circumstances (i.e., regulatory constraints). Constant evolution and analytical-based experimentation with pricing might help to find the unique combination of pricing parameters that will allow the company to reach its objectives and ensure its long-turn market success.

The findings should be considered in light of limitations that may have an impact on generalizability. Our sample of SaaS companies was reasonably limited and not

[2] https://www.g2.com/categories.

randomly selected. Within our study, we felt that we reached a saturation point where the same patterns started recurring, and no new insights were obtained by performing additional interviews. We included several secondary cases to have large, mostly B2C SaaS companies in our sample for analysis. However, a more extensive and more diverse selection of cases may have yielded different findings.

Although this study provides valuable insights into SaaS pricing, we call for further research probing the question of designing and implementing SaaS pricing. Our qualitative study offered a taxonomy of SaaS pricing, but its generalizability is limited. With our previous industry survey [12], this study provides some solid ground for further research that could employ quantitative analyses based on a large industry survey.

References

1. Saltan, A., Smolander, K.: Bridging the state-of-the-art and the state-of-the-practice of SaaS pricing: a multivocal literature review. Inf. Softw. Technol. **133**, 106510 (2021)
2. Laatikainen, G., Ojala, A., Mazhelis, O.: Cloud services pricing models. In: Herzwurm, G., Margaria, T. (eds.) ICSOB 2013. LNBIP, vol. 150, pp. 117–129. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-39336-5_12
3. Lehmann, S., Buxmann, P.: Pricing strategies of software vendors. Bus. Inf. Syst. Eng. **1**, 452–462 (2009)
4. Kullar, P.: 25 Companies Show You Their Best SaaS Pricing Models as Examples. https://blog.upscope.io/25-companies-show-you-their-best-saas-pricing-models/. Accessed 03 Oct 2020
5. Campbell, P.: The Anatomy of SaaS Pricing Strategy (2017)
6. Ojala, A.: Adjusting software revenue and pricing strategies in the era of cloud computing. J. Syst. Softw. **122**, 40–51 (2016). https://doi.org/10.1016/j.jss.2016.08.070
7. Ojala, A.: Selection of the proper revenue and pricing model for SaaS. In: IEEE International Conference on Cloud Computing Technology and Science (CloudCom) Proceedings, pp. 863–868 (2014)
8. Laatikainen, G., Ojala, A.: SaaS architecture and pricing models. In: IEEE International Conference on Services Computing (SCC) Proceedings, pp. 597–604 (2014)
9. Hogan, J., Nagle, T., Hogan, B.J., Nagle, T.: What is strategic pricing? (2005)
10. Kittlaus, H.-B., Fricker, S.A.: Software Product Management: The ISPMA-Compliant Study Guide and Handbook. Springer, Heidelberg (2017). https://doi.org/10.1007/978-3-642-55140-6
11. Yin, R.K.: Case Study Research: Design and Methods. Sage, Thousand Oaks (2009). https://doi.org/10.1097/FCH.0b013e31822dda9e
12. Saltan, A., Smolander, K.: How SaaS companies price their products: insights from an industry study. In: Klotins, E., Wnuk, K. (eds.) ICSOB 2020. LNBIP, vol. 407, pp. 1–13. Springer, Cham (2021). https://doi.org/10.1007/978-3-030-67292-8_1

Is Your Software Ecosystem in Danger? Preventing Ecosystem Death Through Lessons in Ecosystem Health

Robert Evertse, Abel Lencz, Tea Šinik, Slinger Jansen(✉), and Lamia Soussi

Department of Computer Science, Utrecht University, Utrecht, The Netherlands
s.jansen@uu.nl

Abstract. The health of an ecosystem is by definition the most basic requirement for its survival. This paper aims to examine the driving forces behind the health of software ecosystems, in a comparative manner between four different ecosystems which have experienced a major downfall. We examine these ecosystems for similarities, from which demise principles are derived. Consequently, countermeasures are proposed in an attempt to combat these demise principles. The findings show that the main demise principles are *Underestimation of competitors*, *Lack of innovation*, and *Incorrect management of the ecosystem*. The proposed countermeasures to address these demise principles are to *Increase market awareness to increase competitive advantage*, *Increase product quality*, *Increase platform quality*, *Adjust value propositions* and *Formulate a partner-oriented strategy*.

Keywords: Software ecosystem(s) · End-of-life · Software ecosystem demise · Demise principles · Demise countermeasures

1 Introduction

Within a networked environment, the actions of actors cannot be viewed in isolation. The actions have an effect on other parts of the network. A network of organisations can be found in software ecosystems (SECOs). The following definition that will be used throughout this paper: "A software ecosystem is a set of actors functioning as a unit and interacting with a shared market for software and services, together with the relationships among them. These relationships are frequently underpinned by a common technological platform or market and operate through the exchange of information, resources and artifacts" [9]. A SECO is beneficial to all its stakeholders; as customers gain new functionalities, software developers generate revenue and enable lucrative network effects for the SECO's owner. Hence, with the rise and fall of SECOs as more companies adapt their businesses, understanding the mechanism behind the health of such ecosystems has never been so crucial.

Success is not self-evident, as the health of SECOs is volatile to change. There are examples of multiple SECOs that have not managed to survive. Soussi [14]

P. Gregory and P. Kruchten (Eds.): XP 2021 Workshops, LNBIP 426, pp. 96–105, 2021.
https://doi.org/10.1007/978-3-030-88583-0_9

defines software ecosystem death as: "A permanent termination of an entity due to a disturbance in the dynamic between actors where collaborations and links are no longer occurring". Soussi then proposes that the health and death of an ecosystem are influenced by the actors themselves and collaborations between actors. SECO health has been researched earlier [5,6,8,10–12]. E.g., three critical measures for assessing an ecosystem's health are its productivity, robustness, and niche creation [6]. In addition, the main components in the Software Ecosystem Health Framework are actors, software, and orchestration [12]. While current research is mostly focused on measuring SECO health, the academic forum has not yet been able to generalise the flaws in the foundation on which deceased ecosystems were built. The question arises as to which extent the lack of these factors contributes to the demise of SECOs. The goal of this research is to identify elements that negatively affect the health of SECOs. These elements are derived and generalised from examples of discontinued ecosystems, leading to the following research question: ***How can the demise of software ecosystems be prevented?***. The relevance of this research lies foremost in finding explanations of how several SECOs did not manage to survive. In addition, insights are provided on how to address the identified demise principles.

In Sect. 2, the focus lies on four SECOs that have all ceased to exist: BlackBerry OS, Windows Phone, Symbian and Palm OS. For each of the ecosystems, a general outline is created to seek which contributing factors have led them to their demise. The outline is constructed based on the obituaries and eulogies of the SECOs. Year-end reports, blog posts, and other non-academic sources have been used to substantiate the narrative. Consequently, the main contributing factors of the demise is listed per SECO. In Sect. 3, the contributing factors of the demise are specified as well as the countermeasures which SECOs can incorporate as guidelines to not face the same fate. Lastly, in Sect. 4 the main research question will be answered whilst paying attention to the validity of this research and suggestions for future research.

2 Four Case Studies of Demised Software Ecosystems

This section is dedicated to BlackBerry, Windows Phone, Symbian and Palm OS. These cases have been specifically picked due to the similarities shared by the SECOs and to fit a case selection criteria. Firstly, the SECOs were handheld operating systems to enable a simpler comparison of demise factors later on. Secondly, the SECOs have had to achieve a reasonable level of success in the handheld market before experiencing a major decay in their ecosystem health. This is to ensure the availability of data. Lastly, the SECOs have to be end-of-life through official discontinuation.

2.1 Blackberry

BlackBerry Limited, formerly Research in Motion (RIM) used to produce pagers, smartphones and tablets. In 2013, they changed their name to BlackBerry

Limited, hereinafter referred to as BlackBerry. Currently, BlackBerry is a software company specialised in enterprise software.

Background - BlackBerry OS was first used for the BlackBerry 850 two-way pager in 1999. RIM managed to sell multiple phones with a physical keyboard, used mainly for business purposes. The keyboard was considered to be useful for sending e-mails and staying connected. RIM's peak worldwide market share was around 20%.

BlackBerry World reached 60.000 available apps in February 2012. This increased to 90.000 in August of 2012. The estimation is that BlackBerry World reached more than 200.000 available apps. The question arises how such a leading manufacturer did not maintain its market position. According to their respective year-end reports[1], from 2010 onwards the share of handheld revenue of RIM's total revenue started declining. Although the sales show an increasing amount until 2011, RIM has put more focus to their other businesses. Currently, there are several BlackBerry smartphones being sold with Android as their operating system.

Inadequacy 1: Competition with Apple - There are indicators that the rise of Apple and their iPhones was a main contributor to the demise of BlackBerry [13]. Being screen-only devices, they were distinctive from other mobile phones. "*They all have these keyboards, and they are there whether you need them or not to be there*", said Steve Jobs explaining that about 40% of the size of the device was always occupied by the physical keyboard during the iPhone announcement in 2007. BlackBerry deemed Apple yet another competitor into the smartphone market, and was not considered a threat to RIM's core business [13]. We can conclude that one of the inadequacies of the SECO lies in its attitude towards the market. Its underestimation of Apple has played a major role in BlackBerry's demise.

Inadequacy 2: Innovation - BlackBerry adapted the use of a touch screen fairly late. They considered the touch screen to be inferior to a capacitive keyboard and stuck to their familiar design. As years passed by, Apple grew bigger and bigger and even surpassed BlackBerry's sales with the release after the iPhone 4 in 2010. BlackBerry did not timely compete with innovations such as the front-facing camera. By disregarding innovative features for their devices, we can state that innovation was one of the factors why customers switched to different operating systems.

2.2 Windows Phone

The Windows Phone can be seen as the successor of Windows Mobile, and was released on November 8, 2010. It can be considered as a reboot of the Windows Mobile, which used the Windows CE kernel, dedicated to devices with minimal memory. Windows Mobile was first branded as Pocket PC, but from 2003 on, Windows Mobile was the denominator. It was possible to install messaging services, streaming capabilities and even versions of the Office Suite.

[1] https://www.annualreports.com/Company/BlackBerry-Ltd.

Background - Whereas Windows Mobile was more focused to the enterprise market, the Windows Phone had consumers as a target audience. Four versions have been released, namely Windows Phone 7, 8, 8.1 and Windows 10 Mobile. Samsung, Sony Ericsson and LG have produced devices running on Windows Phone, among many other manufacturers. The most notable manufacturer was Nokia, which has led to a partnership between Nokia and Microsoft in 2011. Windows Phone became the primary OS for Nokia devices, which were mostly running on Symbian. The goal of this partnership was to compete with Android and iOS, which were the major mobile SECOs. In 2013, Microsoft acquired Nokia's mobile phone division, aimed to accelerate the growth of Microsoft's share in the smartphone market. However, Microsoft gave up on Windows Mobile in 2017 due to its low market share.

The application store of Microsoft, Windows Phone Store, reached the 100.000 available applications after 20 months in June 2012. It had already surpassed BlackBerry, but was still way behind iOS and Android. They reached the 300.000 milestone in June 2014, whilst Apple's App Store had surpassed the 1.2 million apps. Still, Windows Phone held a mere market share of 3%, which showed how far Microsoft was behind Google and Apple.

Inadequacy 1: Market Entry - Microsoft entered the smartphone market two years after Google's Android was released, and three years after Apple released their first iPhone. The market was starting to grow towards a duopoly, with BlackBerry and Symbian rapidly losing their market share. Customers and developers started committing to either iOS or Android, and their commitment was not as flexible as Microsoft assumed it would be.

Inadequacy 2: Ecosystem Configuration - Microsoft's late release caused developers to be already invested in iOS and Android. The developer interest was relatively low, and the revenue from advertisements was less compared to iOS, Android, BlackBerry and even Symbian, which did harm the enthusiasm of potential developers. This resulted in a lower amount of apps being developed. The appeal of Windows Mobile was not high, leading to a small user base. This also contributes to a lower developer appeal, which results in a vicious cycle. Subsequently, popular applications such as YouTube were not available.

Inadequacy 3: Dependencies - Microsoft adopted Apple's idea of a closed system, trying to maximise control over the developers. However, in the early stages they were dependent on other manufacturers for the hardware. Google's Android, which was released 2 years before, was also dependent on others to produce the hardware, but was a less closed platform. Microsoft's acquisition of Nokia's mobile devices division in an attempt to gain market share and break the duopoly, but this was without success. After writing off the assets, Microsoft discontinued Windows Mobile in 2017.

2.3 Symbian

Background - Symbian was established in 1998, as a partnership between Nokia, Ericsson, Sony, Motorola and Psion. Symbian hoped to achieve similar results as Android by opening up the source code, however this did not save the platform. With the rise of Apple and Android, all (potential) sponsors stopped funding the Symbian Foundation and discontinued their partnership, leaving only Nokia to support the Symbian Foundation and its R&D department. In February 2010, the Symbian Foundation released their code, and shortly after that Nokia announced that it would completely phase out Symbian in February 2011.

Inadequacy 1: Incorrect Management of the Ecosystem - Symbian faced challenges whilst building their ecosystem; they attracted a number of potential ecosystem members who all had their own motives for joining. Symbian transferred its knowledge to partners in three ways: (1) personalized technical support, (2) codified documentation, and (3) Symbian's source code. Most partners only had access to a subset of the source code, because 76.67% of the ecosystem members were part of a competing (mobile phone) ecosystem. To battle these divided loyalties, Symbian introduced the 'refrigeration period'. During a six-month period, the ecosystem partners were prohibited from working for competing platforms. Brusoni and Prencipe [3] mentioned that due to loose coupling the entire ecosystem suffered from poor execution by one or multiple key partners.

Inadequacy 2: Complexity of the Architecture and Code - According to app developers, the problem lies within the user interface. The application developers were faced with several challenge[2]. The first challenge was caused by the customisable design of the OS, the different available versions led to fragmentation, causing some software and apps to be incompatible. The second challenge was that the programming model used by Symbian (in particular memory management) was unlike the models used by iPhone OS, Android or Windows Mobile [9]. Lastly, the third challenge was a result of divided platform control between Symbian and the UI companies. This resulted in developers encountering in hick-ups whilst trying to find documentation.

Inadequacy 3: Weak Defensibility Due to Lack of Resources - In 2008, Symbian and its platform had attracted approximately 9,300 third-party software applications and had been shipped in almost 200 million phones. Even though Symbian managed to acquire a 49.3% of the worldwide smartphone market, it still suffered from great losses. This was caused by the enormous investments into R&D, which were estimated to be more than 200 million from 1998 to 2004. Symbian managed to achieve its first profit in 2005.

Symbian faced difficulties whilst trying to penetrate the North American market they only managed to acquire 3% of the smartphone sales. The entry was predominantly blocked by three local platforms: iPhone (2007), Android (2008) and BlackBerry (2002). Furthermore, the lack of financial resources limited the continuation of projects. An example is that Symbian decided not to proceed in creating its own app store in 2005, which is three years before Apple released the iPhone App Store.

[2] The complexity of the code, increased the average time spent on writing code and thus increased the costs. Development took Nokia approximately 22 months compared to the 12 months it took Windows Phone.

2.4 Palm OS

Background - Out of the four different operating systems examined in this research, Palm OS is the first one to be highly commercially successful in the handheld market [1]. The initial success in the PDA market of Palm OS can be attributed to its user-friendliness and ease of synchronisation with desktop computer. The company created a device and OS that complimented PCs instead of attempting to replace them [4].

Inadequacy 1: Management - Palm Inc. spent $30 million to purchase the Palm trademark and another $44 million for the keystone Palm software. Whilst trying to ease the split between the product's hardware and software components, the company was not able to remain innovative next to the shifting market trends and the increasing pressure from competitors such as Blackberry or Windows Mobile.

After the failed product launch of HP Touchpad (running webOS), the Palm hardware division was discontinued. As a last effort, the webOS software was made open source but it did not halt the mass departure of key engineers working for Palm. Without a strong backbone of a software team, webOS lost its strategic heading.

Inadequacy 2: Innovation - PDA market sank quicker than Palm anticipated as the first smartphones were already sold in the early 2000s. These advanced mobile phones incorporated the same features that PDAs had. By the time Palm reacted to the changing market trends by adding voice and improved data capability to their new devices, the damage could be seen in its market share, dropping from 10% in 2005 to 5% in 2006. Palm was unable to transfer their success from PDAs to smartphones. Palm OS 5 was released in 2002 and was shipped with Palm devices until 2007. During these five years, the OS was drained, receiving no major updates and rapidly aged in terms of design and functionality compared to other operating systems. In addition to being outdated, the Palm OS 5 was suffering from software issues. Palm was in a vicious circle, dropping sales resulted in less revenue and hence lower budget for R&D, however without an up-to-date OS it could not claim a major position in the growing smartphone market.

Inadequacy 3: Ecosystem - Several OEMs (Original Equipment Manufacturer) sold Palm OS based devices however Palm itself viewed the other companies as direct competitors who will reduce their revenue, and hence did not manage to establish long-term and healthy relationships. Furthermore, Android devices exploded in the market in 2009 with 50 different devices sold the first year, made popular among OEMs due to its open-source license. In comparison, Palm OS with its proprietary license was virtually non-existent among OEMs. Palm Pre looked promising however HP was only able to sell the phone initially through Sprint. This hurt sales of the phone as Sprint was third in the phone carrier market and was financially struggling at that time. HP branded Palm Pre Plus also had issues with carriers, namely Verizon refusing a shipment and the negative advertising the device received. The applications for webOS were extremely limited in comparison to Apple or Android. A lack of applications led to fewer users adopting the platform, and fewer users meant that fewer developers produced applications.

Table 1 shows an overview of the SECO characteristics and the applicable demise principles derived from the narrative. Notably, the demise principles do not stand on their own, seeing as their presence enforces one another.

Table 1. SECO overview

	BlackBerry	Windows Phone	Symbian	PalmOS
OS Birth year	1999	2010	1998	1996
OS Latest update	2018	2020	2012	2007
Estimated peak worldwide smartphone market share	22% (2009)	4% (2012)	73% (2006)	10% (2005)
Competitor underestimation	✓	✓		✓
Late market entry		✓		
Lack of innovation	✓			✓
Incorrect management of the ecosystem		✓	✓	✓
Complexity of the architecture and code			✓	
Lack of resources			✓	
Unsupportive partnerships				✓

3 Interpretation: Demise Principles and Countermeasures

Section 2 consists of a detailed research of the four SECOs regarding their background, weak points and the contributing factors which had led to their demise. For each SECO the contributing factors of the demise are specified as well as the countermeasures which SECOs can utilize as a guideline to not face the same fate.

As illustrated earlier, there are some commonalities between the different SECOs and how they did not manage to survive which is listed in Table 2. The commonalities are phrased as demise principles, which are derived from the sources of undesired situations which these SECOs have encountered. Similar sources have been deduced to general SECO characteristics.

We conclude that the different demise principles are not standing on their own, as the demise cannot be attributed to one sole factor. Some factors are tied to each other or can be consequential of one another, as competitor underestimation is some sort of market underestimation, which may result in lower sales and therefore may cause financial issues.

Table 2. Contributing factors to the demise of the observed SECOs.

Demise principles	Countermeasures
Competitor underestimation	Increase market awareness to increase competitive advantage
Late market entry	Increase market awareness to increase competitive advantage
Lack of innovation	Increase product quality & Adjust value proposition
Incorrect management of the ecosystem (including incorrect ecosystem configuration)	Increase product quality & Increase platform quality & Formulate a partner-oriented strategy
Complexity of the architecture and code	Increase product quality & Formulate a partner-oriented strategy
Lack of resources (resulting in a weak defensibility)	Formulate a partner-oriented strategy
Unsupportive partnerships	Formulate a partner-oriented strategy

Formulate a Partner-oriented Strategy - The partner ecosystems are responsible for a significant share of value creation within the ecosystem. There are several ways to examine the partner ecosystem. Lessons can be learned from Avila and Terzidis [2] and Jansen [7] among others. It is important to carefully assess partners, their health and network.

Increase Product Quality - This research illustrated that some SECOs have designed their OS to be too customisable, which led to fragmentation and resulted in incompatible software and apps. Secondly, due to the complexity of the code, the SECOs would spend too much of their resources such as manpower (time) and finances, resulting in a lack of available resources to be allocated for innovation. Possible actions could be to (1) initiate code checks, (2) highlight the bugs and crashes of the code after release updates, (3) evaluations of the accessibility, compatibility and maintainability of the code and User Interface and (4) make use of the available internal and external developer community. By making use of websites such as GitHub and Stack Overflow, the ecosystem can easily monitor which questions are being raised. To increase the community feeling within the developers, SECOs could organize workshops, hackathons and seminars. This will not only promote your product, but it may also lead to innovation which may decrease the development time.

Increase Market Awareness to Increase Competitive Advantages - During the "birth" and whilst growing the ecosystem, it is of importance to perform a market analysis in which the competition is researched. The ecosystem should be constantly aware of its competitors and of their innovations, in order for them to be able to respond promptly and to not lose its customer base or market share. Business and market oriented research has provided several methods to perform market analysis.

Increase Platform Quality - Developers can either make or break a platform. This should force the platforms to create an attractive environment for the developers. There are several ways for platforms to build their developer program. Organisations should pay attention to their API, SDK, documentation among other features. Developer Economics, the largest developer research program across the globe states that platforms could improve the quality of their platform by opting for: access to rich APIs and features, community support, ease of coding and prototyping, low cost development, familiar development environment, revenue potential, good documentation and tech support and large installed base of devices.

Adjust Value Proposition - To address customer needs, one should carefully look at the value ones product portfolio has. USPs (unique selling propositions) are ways of a product to differentiate from other substantiating and competing products. Customer needs are flexible, and change over time. Ideally, an organisation should therefore adapt to the changing customer needs and ensure that the products are continuously developing and adapting to contemporary wishes.

4 Conclusion and Future Work

By analysing four SECOs involved in handheld operating systems which have experienced a downfall, demise principles have been identified. These demise principles with their respective countermeasures are displayed in Table 2. Ideally, by correctly applying these countermeasures, SECOs should be able to thrive.

The four investigated SECOs were similar in nature and market, being active handheld operating systems with a reasonable level of (potential) success. The question arises whether the handheld operating systems market can be compared to other markets targeted by SECOs, and whether the different demise principles and countermeasures are applicable to those. Whether the identified demise principles and countermeasures are correct remains uncertain. They may not be an accurate representation on their own, as they reinforce one another. In addition, other markets may show different demise principles, which could require different countermeasures to correctly solve them.

The authors' recommendation lies foremost in expanding this research, both in breadth as in depth. Including other categories of SECOs could further discover the drive behind SECO demise, and in-depth interviews could help substantiate the claims made in this research. Additionally, a study of a healthy ecosystem where the absence of these demises is observed, could further enforce our hypothesis. Lastly, the proposed countermeasures are hypothetical, and it is important to identify whether the proposed countermeasures are effective. The countermeasures could be applied in unhealthy ecosystems to turn the tide.

References

1. Allen, J.P.: Redefining the network: enrollment strategies in the PDA industry. Inf. Technol. People **17**(2), 171–185 (2004)

2. Avila, A., Terzidis, O.: Management of partner ecosystems in the enterprise software industry. In: International Workshop on Software Ecosystems (IWSECO), pp. 39–55 (2016)
3. Brusoni, S., Prencipe, A.: The organization of innovation in ecosystems: problem framing, problem solving, and patterns of coupling. In: Collaboration and Competition in Business Ecosystems. Emerald Group Publishing Limited (2013)
4. Butter, A., Pogue, D.: Piloting Palm: The Inside Story of Palm, Handspring, and the Birth of the Billion-Dollar Handheld Industry. Wiley, Hoboken (2002)
5. Iansiti, M., Levien, R.: The Keystone Advantage: What the New Dynamics of Business Ecosystems Mean for Strategy, Innovation, and Sustainability. Harvard Business Press, Cambridge (2004)
6. Iansiti, M., Levien, R.: Strategy as ecology. Harvard Bus. Rev. **82**(3), 68–78 (2004)
7. Jansen, S.: A focus area maturity model for software ecosystem governance. Inf. Softw. Technol. **118**, 106219 (2020)
8. Jansen, S., Cusumano, M.A.: Defining software ecosystems: a survey of software platforms and business network governance. In: Software Ecosystems. Edward Elgar Publishing (2013)
9. Jansen, S., Cusumano, M.A., Brinkkemper, S.: Software Ecosystems: Analyzing and Managing Business Networks in the Software Industry. Edward Elgar Publishing, Cheltenham (2013)
10. Mageau, M.T.: The development and initial testing of a quantitative assessment of ecosystem health. Ecosyst. Health **1**, 201–213 (1995)
11. Manikas, K., Hansen, K.M.: Reviewing the health of software ecosystems-a conceptual framework proposal. In: Proceedings of the 5th International Workshop on Software Ecosystems (IWSECO), pp. 33–44. Citeseer (2013)
12. Manikas, K., Hansen, K.M.: Software ecosystems-a systematic literature review. J. Syst. Softw. **86**(5), 1294–1306 (2013)
13. McNish, J.: Losing the Signal : The Untold Story Behind the Extraordinary Rise and Spectacular Fall of Blackberry, 1st edn. Flatiron Books, New York (2015)
14. Soussi, L.: Health vulnerabilities in software ecosystems: five cases of dying platforms. Master's Thesis, Utrecht University (2018)

Genesis of a Wood Harvesting B2B Software Platform

Jaakko Vuolasto(✉) and Kari Smolander

Software Engineering, LUT University, Lappeenranta, Finland
{jaakko.vuolasto,kari.smolander}@lut.fi

Abstract. Digital platform research has focused mostly on global platforms, where the users of the platform are consumers. Business-to-business (B2B) digital platforms have received less attention. This study observes and provides an early report on a digital platform for forestry, bringing together forest companies, contractors, and forest machine manufacturers. The platform started in Finland, but it has begun to extent its scope to international markets as well. We present some early insights about the birth of the platform and the factors that have contributed to its success in the beginning. We also describe some aspects present in B2B platform governance and related forces. Finally, we provide a preliminary outlook of possible future directions of the platform and its ecosystem.

Keywords: Digital platforms · Governance · Platform emergence · Business-to-business platforms

1 Introduction

Digital platforms play an increasingly important role in the everyday life of consumers and businesses. Ghazawneh and Henfridsson [7] define platforms as a codebase providing a set of core functions for modules interoperating with it. A platform becomes more valuable both to its owner and its users when more and more users start using it. This network effect [6] can be direct or indirect.

Platforms can be created with a plan or they can evolve over time. In either case, a successful platform needs a coordinating party, a platform leader. This leadership is about both technology and business [5]. A key task of the leader is to govern the platform, keep it healthy, robust, growing, and offer niches for the parties involved [11]. The key element of governance is decision-making, defining who can make and what decisions [16]. A complementor then provides innovations and solutions that add value to the users of the platform [6].

SDKs, APIs, and related documentation are typical tools that the platform owner provides for application developers. These boundary resources [7] are the interface of the platform for complementors and also users. Boundary resources can be further classified into application, development, and social boundary resources [2].

P. Gregory and P. Kruchten (Eds.): XP 2021 Workshops, LNBIP 426, pp. 106–114, 2021.
https://doi.org/10.1007/978-3-030-88583-0_10

Existing research has various viewpoints to governance. For example, Parker et al. [14] present laws, norms, architecture, and markets of as the elements of platform governance. Ghazawneh and Henfridsson [7] discuss boundary resources as methods of governance. Wareham et al. [18] describe tensions and how they are addressed in ecosystem governance. Wang and Burton-Jones [17] examine how static governance structures and dynamic actions interact and even co-constitute each other. Huber et al. [9] present a process theory of how rules and especially values of an ecosystem affect value co-creation and governance costs. A systematic way of evaluating governance status in different areas and a set of governance practices in the form of a maturity model is provided by Jansen [10].

There is an abundance of studies on global platforms like Apple iOS and Google Android. There are also many studies of business-to-business (B2B) platforms, for example [15, 18]. A key characteristic of platforms in B2B world is separating the ownership of a resource from the value it creates [14]. Another is the global vs. regional perspective; a B2B platform operating in a certain geographic region may face limitations due to the nature of the region or the industry it serves. These local, regional and ownership issues are less studied.

This study approaches these issues with a case study of a digital platform for harvesting and silviculture operations, "Platform" from here on. The Platform connects forest companies (acting as "Service Buyers"), contractors working for them ("Contractors"), and forest machine manufacturers. Service Buyers require stable raw material flow for their factories or business partners. To achieve this, they use Contractors that own the forest machines and operate locally or regionally. Contractors aim to meet the requirements set by Service Buyers and optimize the usage of their forest machines. These two stakeholder groups have tight business relationships and sometimes deviating interests.

The research is in its early phase; this paper delivers some initial results and describes the case. Our research question is the following: what kind of aspects are present in B2B platform governance during its first years. The rest of this paper is structured so that Sect. 2 describes the research design and Sect. 3 the Platform participants and the components of the ecosystem around it. First insights to the research are presented in Sect. 4. Section 5 concludes the paper with an outlook of possible directions for future research.

2 Research Design

The focus is in a single platform, its participants, their interactions, and its governance. While the emphasis of this research is the Platform in Finland, the Platform itself is part of the Software Company's global portfolio.

We used Grounded Theory as the research method [4]. Our aim was to understand the phenomenon in its natural environment, by using multiple sources of information: interviewing platform stakeholders and observing publicly available documentation. We targeted the first round of 29 interviews mainly towards the Software Company, Service Buyers (SB), and Contractors (Table 1). The interviews were semi-structured and performed remotely due to COVID-19 restrictions. The interviews were transcribed, and the transcriptions were analyzed with coding.

Table 1. First round of interviews

Organization	Role
Consulting Firm	Project Manager
Contractor 1: harvesting, large, single SB	Account Manager
Contractor 2: silviculture, large, multiple SBs	Manager
Contractor 3: harvesting, large, multiple SBs	Manager
Contractor 4: harvesting, large, single SB	CEO
Contractor 5: silviculture, small, single SB	CEO
Contractor 6: harvesting, small, single SB	CEO
Contractor 7: harvesting, large, single SB	CEO
Educational Institution	Teacher, Harvesting
Machine manufacturer	Technical Customer Support Manager
Service Buyer A	Senior Vice President, Development
Service Buyer A	ICT Solution Designer
Service Buyer B	System Specialist
Service Buyer C	Development Manager
Service Buyer C	Development Specialist
Service Buyer C	Team Lead, Information Management
Service Buyer D	SVP, Innovation and Development
Service Buyer D	Solution Architect
Service Buyer D	Development Manager, Harvesting
Service Buyer D	Operations Manager
Service Buyer E	Manager
Service Buyer F	Manager
Software Company	Product Owner
Software Company	Service Manager
Software Company	Service Manager
Software Company	Product Owner
Software Company	General Manager
Software Company	Key Account Manager
Wood procurement R&D company	CEO

The observed documentation consisted of two communication standards [19, 20] that provide the foundations for the Platform boundary resources, and a set of regulations created by the Finnish Forest Industry [12] that provides the rules regarding forest machine data ownership and usage.

Grounded theory is about interacting with the data and comparing data, codes, and emerging concepts [3]. In this study the comparison is still in its early phase: we performed the first round of interviews during February and March of 2021. While the analysis is in progress, some of the first emerging concepts are presented here.

3 Platform Description

3.1 Participants

There are about 25 Service Buyers using the Platform in Finland. These include publicly listed large companies, a state-owned enterprise, and Forestry Management Associations. The number of Contractors using the Platform is currently around 800 in Finland. For a Contractor it is mandatory to use the Platform when contracting with a Service Buyer using the Platform. A special subgroup of Contractors are Educational institutions. They train new forest machine drivers and use the Platform as part of the training.

Software Company is the platform leader. Additionally, it is also a Platform Complementor while developing systems for the Service Buyers using the Platform.

Forest Machine Manufacturers make harvesters and forwarders. A harvester is used to fell and cut trees in preferred lengths. A forwarder is used to transport the logs to an intermediate storage. They have control systems that are integrated to the Platform. About ten manufacturers have presence in the Finnish market.

Platform Complementors develop solutions for the Platform. This refers to the makers of the information systems of Service Buyers or control systems of forest machines.

There are two special organizations in the ecosystem. Finnish Forest Centre[1] is a state-funded organization that collects and provides data about the forests in Finland, advices forest owners, and enforces forestry legislation. It has a central role in forestry related information systems in Finland, as it provides the Finnish Forest Data Standards [19], open data sets, APIs, and related tools.

Metsäteho[2] is a R&D company owned by forest industry organizations Finland. It has coordinated the creation of recommendation on the ownership, use and processing of data in forest machines [12]. It is also the Finnish coordinator of the StanForD standard [20].

3.2 Components of the Ecosystem

In the core of the Platform are features for planning, executing, and reporting harvesting and silviculture operations. The features are divided into client applications, common services for them, and an integration solution. Software Company has retained the development of all client applications to itself, while providing an API for other ecosystem participants, utilizing the Finnish Forest Data Standards [19]. Consumers of this API are the ERP systems of the Service Buyers.

[1] https://www.metsakeskus.fi/en.

[2] https://www.metsateho.fi/briefly-in-english/.

ERP systems of Service Buyers tap into the Platform using the API mentioned above. There are Service Buyers whose ERP is provided by the same Software Company that is the Platform Owner. Additionally, there are Service Buyers in the Ecosystem, whose ERP is provided by another vendor. These complementing vendors were not interviewed in the first round.

Each forest machine manufacturer has a control system that steers the hardware of the machine. This control system interacts with the Platform client using the StanForD standard [20], either a text based older version or a more recent XML version. Both use files for information exchange.

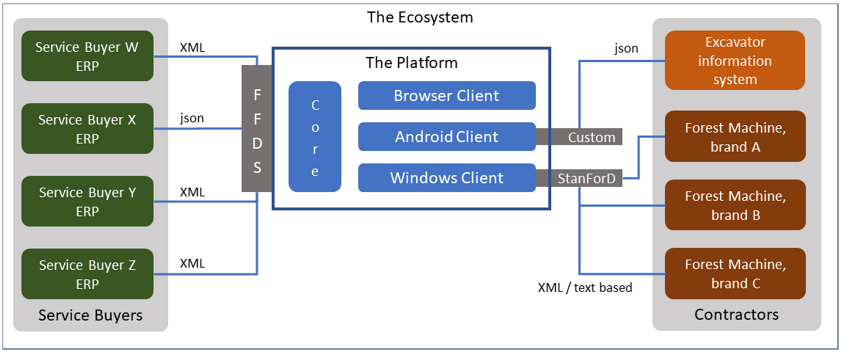


Fig. 1. The Platform and the ecosystem around it

The components of the ecosystem are presented in Fig. 1. Integrations with Service Buyer ERPs and forest machine control systems were implemented already during the initial implementation project. A more recent addition is the integration with an excavator information system used in planting of seedlings. The API based on Finnish Forest Data Services and the StanForD file-based integration are currently the boundary resources that the Platform offers. An example of a social boundary resource is a regular meeting with the Machine Manufacturers, organized by the Software Company.

4 First Insights

4.1 Platform Genesis

The Platform started out as a joint project of three Service Buyers to optimize their business processes with contractors. The companies created a common requirements specification and through a public procurement chose the Software Company for design, implementation and running the service.

Design specifications of the Platform were written in 2013 and the implementation project started in 2014. The first production deployments were in 2016, and production use gradually expanded during the following years, so that by 2019 all of the three founding Service Buyers had their harvesting and silviculture operations running on the Platform.

The first years of the Platform have been a success. Most of the interviewees saw that the Platform has fulfilled its purpose well or at least reasonably. The use of the Platform has expanded both in Finland and internationally. While it is obligatory for a Contractor of Service Buyers to use the Platform, several interviewed Contractors mentioned that they would use it even if it was optional. The Platform has replaced the previous separate Service Buyer specific systems with a single solution. This has helped in optimizing the use of heavy machinery. For instance, Contractors are able to plan and execute their work so that machines operating in a certain area can now be used in working sites for different Service Buyers, which was very difficult or even impossible with the previous separate systems.

The founders had a common interest in creating an outsourced service. Although the founding Service Buyers are also competitors, they saw the benefits in developing a solution to a pre-competitive area. Instead of trying to create an industry-wide solution at once, two founding Service Buyers agreed first with each other and then a third one joined the venture. It also helped that the information systems of Service Buyers serving the same purposes as the Platform required renewal.

There were communication standards [19, 20] in place that helped creating the boundary resources to ERP and control systems. The recommendations about the forest machine data [12] and guidelines for following competitive legislation – created by the founders – can be viewed as examples of self-governance in [8]. Finally, the founders had positive experiences from a similar platform in wood logistics, also developed and run by the Software Company.

The founding Service Buyers had a strong role in the beginning. Having created the requirement specifications, they steered the development and governed the emerging ecosystem. As part of the agreement there is a framework for governance. It specifies rules for common development, decision-making structures, and for example the service level agreement in Finland. It was initially presented in 2013 as a part of the procurement documents. It was further developed during the implementation project and tested both in the first deployments and during the first years in production.

4.2 Current Aspects to Governance

Joining the Platform is rather straightforward. For a Contractor to enter the Platform two agreements are required: first with a Service Buyer about the contracting work, and second with the Software Company about using the Platform. A new Service Buyer can join the Platform by a negotiation with the Software Company. In this process it must be made sure that the conditions specified in the guidelines are met.

Adding new features to the Platform is more problematic. It was mentioned in the interviews that the needs of the Contractors should be the major driver of the Platform development. It seems that the requirements coming from Contractors are not getting through as well as expected. At least the following aspects have surfaced during the initial analysis.

Service Buyers and Contractors. The founding Service Buyers are established large companies with IT departments and routines for the customer-supplier dialogue. When they need a new feature on the Platform, they are more qualified to argue for their case.

At the same time the Contractors are a heterogenous group. One-man company or a company having a fleet of 30 machines can have very different interests and abilities to influence the development of the Platform. The Contractors are the largest user group, but they represent the minority in the decision-making structures. The Software Company is expected to act as a balancer.

Slow Development. Day-to-day operation of the Platform is considered stable and satisfactory. However, development of new features or improvements in the Platform receives critique. Compared to exclusive software development, single vendor for a single customer, Service Buyers saw the Platform development progressing slower. Complexity – meaning a greater number of participants – was identified as one reason, but the shortage of development resources of the Software Company was mentioned as well.

Finland and International. Although the focus of interviews was in Finland, the international aspect came also up. This is natural, as the potential market for the Platform in Finland is limited by the amount of Service Buyers and Contractors. It is one of the features of this B2B market: Contractor operates in a certain region and has a limited number of Service Buyers as its customers. Although the Software Company recognized the potential of the international market already in the very beginning, the implementation has not been without problems. There are differences in requirements and business processes in different geographic regions.

4.3 Looking Ahead: Role of Complementors

The number and types of Complementors are currently somewhat limited. In spite of that, the Software Company has clearly the leadership as described in [13] and it has the critical mass in Finland. The possible directions from here are interesting: as the regional market has its limits, geographic expansion is one way, which the Software Company has already pursued. Another possibility is to open the ecosystem more, which means emphasizing the role of Complementors. This will be a strategic decision for the Software Company – while growing, it needs to maintain control and protect its business interests. Understanding the value creation came up often in the interviews of the Software Company, as a prerequisite to introducing new Complementors to the ecosystem. However, controlling the ecosystem too tightly can result in lack of generativity and innovations, as reported in [1].

5 Conclusions

The case provides a classical setting: a platform leader balancing governance actions. The Service Buyers had a strong role in the beginning – a power shift to Software Company is well on its way, but perhaps not yet complete. A key issue is hearing the voice of Contractors.

Observing and analyzing B2B digital platform governance can help understanding the transformation of the industry. As timber markets, actors, and their interplay are not the same everywhere, geographic expansion is not a trivial option. Introducing more complementors to an existing ecosystem is not easy either.

The analysis of the results of the first interviews will continue, combined with a second targeted round towards Contractors and Complementors. A potential future direction for research is the role of the Software Company as both the leader and a Complementor; what kind of scenarios are possible with the current setting and with opening the ecosystem more. Data economy as an enabler for ecosystem self-renewal deserves attention. Also, a possible research avenue is in demarcation within the ecosystem: what is developed in the core of the Platform and what kind of features are implemented in the periphery.

References

1. Bazarhanova, A., et al.: Love and hate relationships in a platform ecosystem: a case of Finnish electronic identity management. Presented at the January 3 (2018). https://doi.org/10.24251/HICSS.2018.187
2. Bianco, V.D., et al.: The role of platform boundary resources in software ecosystems: a case study. In: 2014 IEEE/IFIP Conference on Software Architecture, pp. 11–20 (2014). https://doi.org/10.1109/WICSA.2014.41
3. Charmaz, K.: Grounded theory methods in social justice research. In: The SAGE Handbook of Qualitative Research, pp. 359–380. SAGE (2011)
4. Corbin, J.: Basics of Qualitative Research: Techniques and Procedures for Developing Grounded Theory. SAGE, Los Angeles (2015)
5. Gawer, A., Cusumano, M.: How companies become platform leaders. MIT Sloan Manag. Rev. **49**(2), 28–35 (2008)
6. Gawer, A., Cusumano, M.A.: Industry platforms and ecosystem innovation. J. Prod. Innov. Manag. **31**(3), 417–433 (2014). https://doi.org/10.1111/jpim.12105
7. Ghazawneh, A., Henfridsson, O.: Balancing platform control and external contribution in third-party development: the boundary resources model. Inf. Syst. J. **23**(2), 173–192 (2013). https://doi.org/10.1111/j.1365-2575.2012.00406.x
8. Gorwa, R.: What is platform governance? Inf. Commun. Soc. **22**(6), 854–871 (2019). https://doi.org/10.1080/1369118X.2019.1573914
9. Huber, T.L., et al.: Governance practices in platform ecosystems: navigating tensions between cocreated value and governance costs. Inf. Syst. Res. **28**(3), 563–584 (2017). https://doi.org/10.1287/isre.2017.0701
10. Jansen, S.: A focus area maturity model for software ecosystem governance. Inf. Softw. Technol. **118**, 106219 (2020). https://doi.org/10.1016/j.infsof.2019.106219
11. Jansen, S., Cusumano, M.: Defining software ecosystems: a survey of software platforms and business network governance. In: Software Ecosystems: Analyzing and Managing Business Networks in the Software Industry, vol. 879, (2013). https://doi.org/10.4337/9781781955628.00008
12. Metsäteho: Metsäkonetiedon omistusta, käyttöä ja käsittelyä koskevat periaatteet. http://www.metsateho.fi/metsakonetieto-suositus/. Accessed 14 Oct 2020
13. Moore, J.F.: Predators and Prey: A New Ecology of Competition (1993). https://hbr.org/1993/05/predators-and-prey-a-new-ecology-of-competition
14. Parker, G.G., et al.: Platform Revolution: How Networked Markets Are Transforming the Economy and How to Make Them Work for You. W. W. Norton & Company, New York (2016)

15. Shestakofsky, B., Kelkar, S.: Making platforms work: relationship labor and the management of publics. Theory Soc. **49**(5–6), 863–896 (2020). https://doi.org/10.1007/s11186-020-09407-z
16. Tiwana, A., et al.: Research commentary—Platform evolution: coevolution of platform architecture, governance, and environmental dynamics. Inf. Syst. Res. **21**(4), 675–687 (2010). https://doi.org/10.1287/isre.1100.0323
17. Wang, G., Burton-Jones, A.: Rethinking IT governance structure and action. In: ICIS 2020 Proceedings (2020)
18. Wareham, J., et al.: Technology ecosystem governance. Organ. Sci. **25**(4), 1195–1215 (2014). https://doi.org/10.1287/orsc.2014.0895
19. Finnish Forest Data Standards. https://www.metsakeskus.fi/fi/avoin-metsa-ja-luontotieto/suorakayttoaineistot/metsatietostandardit. Accessed 09 Apr 2021
20. StanForD. https://www.skogforsk.se:443/english/projects/stanford/. Accessed 10 Apr 2021

Towards a Taxonomy of Impact Factors for Digital Platform Pricing

Virginia Springer(✉) and Dimitri Petrik

Department VIII: Information Systems II, University of Stuttgart, Keplerstr. 17,
70174 Stuttgart, Germany
springervirginia@googlemail.com,
dimitri.petrik@bwi.uni-stuttgart.de

Abstract. Solving the chicken-or-egg problem and leveraging value contributing actors on the platform is crucial to establish dynamic platform-based ecosystems. A digital platform provider is challenged to manage multilateral platform architecture and governance mechanisms to establish an attractive platform-based ecosystem to foster third-party complementors to join. One of the key issues while establishing a platform-based ecosystem remains the decision about an adequate pricing model. Despite a large number of publications on platform governance, detailed pricing model analyses remain rare. In this explorative paper, we conduct a literature review, studying 62 relevant papers to explore the pricing impact factors to create a foundation for future research of price models in the under-researched setting of the Industrial Internet of Things (IIoT). The most relevant pricing factors and their distinctive characteristics are summed up in a multi-dimensional taxonomy. The developed taxonomy includes 13 impact factors and 38 characteristics of platform pricing. Our findings enable the decomposition and understanding of price models for their future improvement.

Keywords: Platform pricing · Pricing impact factors · Pricing taxonomy · IIoT platforms · IIoT platform pricing · Literature Review

1 Introduction

The Internet of Things (IoT) and its industrial area of application (IIoT) build a top-priority topic in the digitization of products and processes. Digitized products erode the established boundaries between the companies, e.g., by making the customer data available for sharing and processing, ultimately leveraging value-added services [1, 2]. Industrial companies rely on digital industrial platforms, known as IIoT platforms, since they operate as scalable middleware systems in digital infrastructure, integrating networked subsystems and heterogenous third parties in the value creation process [3, 4]. Besides, platforms offer a stable set of functionalities as an extensible technological foundation for modular innovations to be built upon. Since the services can be co-created by multiple actors, platforms also organize the interaction in an ecosystem, providing a transactional base. Hence, IIoT platforms correspond simultaneously to the innovation and transaction platform concepts [4, 5]. Similarly, indirect network effects and generativity are recognized positive effects of platformization, requiring an innovation-contributing ecosystem

P. Gregory and P. Kruchten (Eds.): XP 2021 Workshops, LNBIP 426, pp. 115–124, 2021.
https://doi.org/10.1007/978-3-030-88583-0_11

of platform users [5]. Following the idea of ecosystem development, it is in the platform providers' interest to shift industrial supply relationships into platform-based transactions and transform the supply chains into ecosystem participants [6]. The operation of digital platforms incurs costs, and apart from the initial development of the platform core, the variable cost of attracting new platform users can be high. This is caused by subsidizing the actors, as one of the known platform launch strategies is to subsidize certain user segments to engage them to join the ecosystem and solve the "chicken-egg-dilemma" [7–9]. In the IIoT, attracting new actors is an even bigger challenge due to the different functioning of indirect network effects, the non-standardized business relationships, and the variety of complementary partner types [4, 6]. However, most IIoT platforms on the market charge different platform-ecosystem participation fees, pursuing different pricing strategies [10]. Pricing strategies are understood as strategic approaches that "*a firm adopts to determine what it will charge for its products and services* [11]". In this context, pricing impact factors combine designable parameters and exogenous characteristics (i.e., platform economics), determining the pricing strategy. Although the platform research agrees on the importance of pricing models in ecosystem development and categorizes pricing as a relevant pillar in software product strategy [12], there are only a few research works [9, 13] that offer in-depth analysis of the existing price strategies and impact factors in the context of platform-ecosystems. Inspired by this state of research, our research goal for this preliminary paper is to *identify the price impact factors relevant for the pricing strategy of digital platforms* by conducting a literature review and discovering suitable price impact factors from the broad research field on digital platforms.

2 Theoretical Background

The IoT paradigm integrates technology-enabled physical objects into a global cyber-physical network that changes how to create added value. The concepts of IoT and IIoT are often equated in theory due to their (technological) similarity. However, multiple characteristics differ depending on the end-users, the industry focus, the underlying service models, and the connected things. IIoT supports the emergence of digital and smart manufacturing, aiming to integrate operation technology (OT) and IT domains to create economic value. Therefore, industrial organizations are usually considered the primary end-users. Thus, the IIoT aims to connect all industrial assets, including machines and control systems, intra-organizational information systems (IS), and business processes in a B2B environment [4, 14]. IIoT platforms can be seen as middleware systems that orchestrate the heterogeneous landscape of connected assets and software systems. Hence, the IIoT platform usually provides a technological infrastructure fostering connectivity and interoperability between intelligent machines, control systems, and software systems [3, 6, 14]. On top of the technological infrastructure, applications enable data-driven services to the platform users. While traditional pipeline companies operate within corporate boundaries, platform providers use an ecosystem of autonomous parties to create shared value and exploit the potentials of generativity [15]. From a technical perspective, platforms offer an extensible technological foundation on top of which third parties can build value-adding applications [13]. From an organizational view, platforms act as multi-sided markets, acting between several user segments to bring them together in an overarching ecosystem [16]. Fostering users to join a platform-based ecosystem, platform providers can operate different mechanisms within the architecture and governance fields of action [13]. Pricing is recognized as one of the governance mechanisms and a platform launch strategy since a platform provider can vary the price model between different market sides and subsidize specific user groups [7]. Besides, a platform provider can define the revenue sharing model to foster the complementors' innovation activities [13]. Preliminary work considers multiple connection points between the ecosystem participants and IoT platforms that can be charged for [17]. However, only a few studies offer a holistic overview of the existing platform price impact factors, especially considering the enterprise instantiation of the platform in the IIoT. As Schreieck et al. already mentioned, the price model for IoT platforms may depend on heterogeneous factors determined by the inherent technological complexity [17]. To sum up, the lack of knowledge on the IIoT platform pricing makes a literature review necessary to reduce the complexity in the design of price models regarding their underlying impact factors.

3 Research Methodology

We developed a three-stage research framework to guide the development of a pricing taxonomy. Classification plays an important role, structuring knowledge on a particular object of interest. Since many classification approaches lack a profound methodology, we consequently adopted our overall procedure to the widely used iterative taxonomy development approach presented by Nickerson et al. [18]. We opted for Nickersons' approach since it is already evaluated and established in the IS research [19, 20].

According to the taxonomy development procedure, we identified meta-characteristics as well as ending conditions first. Meta-characteristics help to define the purpose of the taxonomy and address the interests of future research. We used the VISOR framework according to [21] to define our meta-characteristics. As already shown in Sect. 2, the characteristics of platform business models differ from traditional business models primarily in terms of the exponential relationship between the value and the number of users of a platform, the value creation in ecosystems, and the ability to interoperate with other services. Since the VISOR framework emphasizes the importance of digital platforms' central role, the need to orchestrate multiple actors in ecosystems, and consider the multitude of interfaces (as customer touchpoints), it seems suitable to align the derived impact factors with the dimensions of the framework. It hence composes digital business models in the five dimensions of Value Proposition, Interface, Service Platforms, Organizing Model, and Revenue Model. In line with Nickerson, the subjective ending conditions aim to ensure comprehensive, extendible, concise, robust, and explanatory results. The ending conditions mainly include specifications on the objects' classification, representativeness of the impact factors, changes in the taxonomy, and the uniqueness of the impact factors and characteristics. Since the purpose of our taxonomy is to provide a representative overview of the pricing impact factors for IIoT platforms in an emerging domain, we followed a conceptual-to-empirical approach.

Following the conceptional-to-empirical approach, we built our initial taxonomy iteratively on existing literature. We screened the titles, abstracts, and – where necessary – full-texts to conduct a rigorous literature analysis, which was summed up in a concept matrix. In particular, literature that refers to pricing or merely mentions it but does not examine it was excluded. Our literature analysis mainly focused on economic models on multi-sided markets and empirical or conceptional studies on platform governance investigating pricing impact factors. We used various keyword combinations like “multi-sided platform pricing” or “platform governance pricing” in multiple iterations to build an initial corpus consisting of 45 papers, building the first iteration (conceptual-to-empirical). With exclusive forward and backward research (second iteration), the corpus included 62 final papers for analysis, building the third iteration (conceptual-to-empirical). The overview of the search details as well as the complete list of papers and a concept matrix are available at the following URL: https://bit.ly/3g4PyG2.

In line with the concept of triangulation, we mainly focused our analysis in the first iteration on economic models on multi-sided markets and identified eight impact factors. Our second iteration that mainly focused on empirical and conceptional studies on platform governance identified three more impact factors. In the third and last iteration, we identified in total 13 pricing impact factors encompassing 38 characteristics. Each dimension is indicated by mutually exclusive (E) or non-exclusive (NE) characteristics.

4 Taxonomy

This section presents our taxonomy systematizing the derived price model impact factors with their characteristics, structured according to the VISOR framework’s dimensions.

Table 1. Taxonomy of Price Impact factors for Digital Platforms

Dimension	Impact factor		Characteristics				E/NE
Value Proposition	P1	Market structure	Monopoly	Duopoly	Oligopoly		E
	P2	Platform differentiation	Platform offer	Network size	Bundling of services		NE
	P3	Platform offer	Specialized platform		Platform offers industry solutions		E
Interface	P4	Platform access	Open		Restricted		E
	P5	Boundary Resources	ABR	DBR	SBR		NE
Service Model	P6	Modularity	Low modularity		High modularity		E
	P7	Platform architecture	Purists	Analysts	Connectors	Allrounder	E
	P8	Platform lifecycle	Emerging	Growth	Maturity	Saturation	E
Organizing Model	P9	Information availability	transparent		Not transparent		E
	P10	Network effects	Indirect positive	Indirect negative	Direct positive	Direct negative	NE
Revenue Model	P11	Pricing model	Subscription-based	Usage-based	One-time-Payment		NE
	P12	Subsidization	Asymmetric (subsidization)		Symmetric (no subsidization)		E
	P13	Pie-Splitting	Fix	sliding	rising	No pie-splitting	E

Value Proposition: This dimension contains the three pricing impact factors market structure (P1), platform differentiation (P2), and platform offer (P3). **P1** is determined by the supply and demand sides’ number and size and describes an economic market’s structure and composition. Existing research focuses primarily on the ideal-typical case of a monopoly and partly on the case of a duopoly. Only in a few investigations, the oligopoly case has been considered yet. Against this background, existing research shows that monopolies generally have greater freedom to set prices and that pricing options may decrease as competition increases [22, 23]. **P2** subsumes those characteristics of an ecosystem that are aimed at the heterogeneity of value creation ecosystems. This non-exclusive impact factor shows how platform providers can differ from one another. We distinguish the three characteristics platform offering, network size, or bundling of services. In this context, recent research shows that the price can significantly depend on the network size [24, 25]. According to research, product differentiation always leads to higher welfare through better demand stimulation, which directly impacts price [26]. Also, bundling of services can lead to higher demand stimulation and thus significantly influence pricing [27]. Whether combinations of these characteristics (e.g., platform offer and bundling) are correlated and how they influence the price have not been investigated yet. Lastly, **P3** distinguishes between platforms that focus on specialization or platforms that offer industry-wide solutions. Against this background, specialization of the platform offering is accompanied by higher demand stimulation for heterogeneous markets since specialized platform providers are less substitutable and can respond better to the specific customer demands [28].

Interface: This dimension contains the impact factors platform access (P4) and boundary resources (BR) (P5). Both are closely related to providing services and access to the underlying platform. The platform access may vary between openness and restrictions set by the platform provider. Against this background, platform openness can significantly influence the platform's attractiveness, directly impacting the price [29, 30]. **P5** indicates that payment barriers for the use of specific BRs may be set up [17]. BRs represent different technical and non-technical resources for third-party access of the platform that can be conceptually divided into application BR (ABR), development (BR), and social BR (SBR) [31]. For instance, a platform provider can charge additional fees for participation in the partner program or support.

Service Platform: This dimension describes the impact factors modularity (P6), platform architecture (P7), and platform lifecycle (P8) that enable, shape, and support the business processes and relationships in platform ecosystems. **P6** subsumes characteristics related to the platform modularity since they are hardly reversible later and, therefore, influence orchestration and aligning price level [13]. **P7** describes how the architecture of platforms may differ, affecting the price due to the different levels of modularity and the added value (i.e., generic development tools or specific use case analyses). For instance, using an IoT platform may result in full-stack services or solely in high-end analytics, which requires distinctive pricing [32]. Lastly, **P8** subsumes those characteristics related to the degree of performance or the degree of exploitation of competitive potential by the platform. Against this background, it is crucial to consider how technologies and technological concepts such as digital platforms reach technical performance limits in their further development potential [13]. Depending on the growth ambition of the platform provider in different phases, subsidization may influence the price.

Organizing Model: This dimension addresses the fundamental organizational model of digital platforms and describes how business processes, value chains, competition, and partner relationships need to be organized. We distinguish the two pricing impact factors, information availability (P9) and network effects (P10). The **P9** subsumes those characteristics that relate to information transparency in the platform ecosystem. Information transparency is crucial for a perfect market and is given when all actors in the ecosystem possess all information about environmental states and their actions. This also includes information about the uncertainty that can arise in a transaction (on a digital platform). The platform operator's influence is indirect since the entire ecosystem determines information availability and related uncertainties. Existing results show that information transparency and the associated uncertainties directly impact price elasticity [33]. According to [34], transparent (price) information leads to users' expectations being better met and thus to the effect of price reductions being strengthened. **P10** summarizes the interdependent relationships between the various actors that determine platform business models' success in ecosystems. The four characteristics, including indirect positive, indirect negative, direct positive, and direct negative, can be distinguished based on the literature analysis. Established research shows that the platform operator's optimal prices depend on how (new) participants influence the attractiveness of the ecosystem [35]. Moreover, pricing should also take into account how strongly participants in the ecosystem respond to network effects in general [36].

Revenue Model: This dimension includes the three impact factors: pricing model (P11), subsidization (P12), and pie-splitting (P13). **P11** subsumes non-exclusive fees that may be incurred by ecosystem actors when using the platform. These are usually linked to licensing and are either subscription-based, usage-based pricing models or a one-time payment. **P12** indicates whether the platform access fees hit all sides of a platform or whether specific sides are subsidized. Accordingly, one can distinguish between symmetric (no subsidization) and asymmetric (subsidizing a market side) revenue models. According to [37], a subsidization interest for platform providers results from indirect network effects. Against this background, [26] show that the side that is more price-sensitive should be subsidized. Consequently, revenues for the platform operator are primarily generated on that side of the market that is more affected by network effects. Therefore, this market side contributes to increased market participation [38]. Lastly, **P13** relates to revenue sharing and associated pie-splitting conditions between the platform provider and the platform users. Typically, complementors can be charged for selling their platform-based solutions. Four types of pie splitting were already recognized: fixed, sliding, rising, and no pie-splitting [13]. Sliding refers to a decreasing revenue share with increasing demand while rising forces an increasing revenue share with increasing demand. Existing research by Rochet and Tirole shows that rising pie-splitting conditions, in particular, are an effective means of increasing the attractiveness of a platform for homogeneous products or services [16]. Although pie-splitting conditions can directly contribute to ecosystem development, especially in the platform life cycle's early phases, blueprints for adequate pie-splitting models for heterogeneous platform domains, such as IIoT, have not yet been considered in the literature.

5 Discussion and Conclusion

This paper used a structured literature review to derive a multidimensional taxonomy of price model impact factors for digital platforms. Our findings have implications for the following three research discourses. Firstly, they contribute to clarifying possible pricing impact factors by enabling a more differentiated understanding of pricing impact factors at various levels of platform business models. Thus, our preliminary work responds to current demands for more transparency and further research in this emerging area and provides a grounding for domain-specific pricing research in the IIoT context. Secondly, the derived impact factors move forward the research on the platform governance and platform establishment strategies [7, 17] since the taxonomy creates transparency for the pricing design and enables platform providers to adjust the prices more delicately. Consequently, the price level for accessing the platform can be adjusted according to the participation level differing between the ecosystem participants and supporting stakeholder-specific platform governance [10]. Besides, our findings may support implementing the so-called "goldilocks" rule considering the platform price level [13]. Thirdly, most of the impact factors were extracted from the literature on multi-sided markets, being studied less from the platform governance research stream. Accordingly, the results build one of the early contributions to transfer the findings from the economic research field to the platform governance stream within the platform research [39] since the platform governance literature primarily classifies pricing as a governance mechanism without going into much detail [17].

From a practical perspective, the taxonomy provides a foundation for pricing decisions and may support platform providers in the creation of a price strategy blueprint. However, regardless of the results obtained, we have to point out the current limitations of the taxonomy. The current state of the taxonomy lacks empirical validation with market-ready IoT platform providers. Therefore, additional empirical-to-conceptual validation of the taxonomy is lacking. Consequently, the taxonomy also does not offer domain-specific archetypes of pricing strategies for IoT platforms. Since there has been little research to date on pricing impact factors in the context of IIoT platforms, it is conceivable that the impact factors and characteristics selected for the taxonomy will apply only conditionally to (future) IIoT platforms. Motivated by these limitations, our future research will focus on empirical validation of the taxonomy, with a subsequent clustering to identify pricing strategies for IIoT platforms. Besides, with the addition of empirical studies, it is to be expected that the taxonomy may need to be adapted, as the IIoT market itself is very dynamic, and so are the associated business models. Therefore, it is likely that as new IIoT platform business models evolve, there may be new pricing impact factors and that impact factors that are currently possibly underrepresented will gain importance and relevance. Therefore, it is necessary to regularly update the taxonomy and extend it accordingly to ensure representative results. Due to this, the taxonomy represents only a first step towards structuring possible pricing impact factors in the context of the IIoT and reveals significant deficits that offer further research opportunities. Besides, the analysis of the existing price impact factors forms the basis for composing a research agenda on platform pricing in the context of platform ecosystems.

References

1. Porter, M., E., Heppelmann, J. E.: How smart connected products are transforming competition. Harvard Bus. Rev. 92(11), 64–8 (2014)
2. Leminen, S., Rajahonka, M., Wendelin, R., Westerlund, M.: Industrial internet of things business models in the machine-to-machine context. Ind. Mark. Manage. **84**, 298–311 (2020)
3. Mineraud, J., Mazhelis, O., Su, X., Tarkoma, S.: A gap analysis of Internet-of-Things platforms. Comput. Commun. **89–90**, 5–16 (2016)
4. Pauli, T., Fielt, E., Matzner, M.: Digital industrial platforms. Bus. Inf. Syst. Eng. **63**(2), 181–190 (2021). https://doi.org/10.1007/s12599-020-00681-w
5. Gawer, A.: Digital platforms' boundaries: the interplay of firm scope, platform sides, and digital interfaces. Long Range Planning (appearing soon)
6. Petrik, D., Herzwurm, G.: Towards the IIoT ecosystem development – understanding the stakeholder perspective. In: Proceedings of the 28th ECIS, pp. 1–17, Marrakesh, Morocco (2020)
7. Stummer, C., Kundisch, D., Decker, R.: Platform launch strategies. Bus. Inf. Syst. Eng. **60**(2), 167–173 (2018). https://doi.org/10.1007/s12599-018-0520-x
8. Kenney, M., Rouvinen, P., Seppälä, T., Zysman, J.: Platforms and industrial change. Ind. Innov. **26**(8), 871–879 (2019)
9. Cusumano, M.A., Gawer, A., Yoffie, D.B.: The Business of Platforms: Strategy in the Age of Digital Competition, Innovation, and Power. HarperBusiness, New York (2019)
10. Wareham, J., Fox, P.B., Cano Giner, J.L.: Technology ecosystem governance. Organ. Sci. **25**(4), 1195–1215 (2014)
11. Sammut-Bonnici, T., Channon, D.F.: Pricing Strategy. Wiley Encyclopedia of Management, New York (2015)
12. Kittlaus, H.-B., Clough, B.: Software Product Management and Pricing. Springer, Berlin-Heidelberg (2009)
13. Tiwana, A.: Platform Ecosystems: Aligning Architecture, Governance, and Strategy. Mor-gan Kaufmann, Amsterdam (2014)
14. Wortmann, F., Flüchter, K.: Internet of things. Bus. Inf. Syst. Eng. **57**(3), 221–224 (2015). https://doi.org/10.1007/s12599-015-0383-3
15. Hein, A., Weking, J., Schreieck, M., Wiesche, M., Böhm, M., Krcmar, H.: Value co-creation practices in business-to-business platform ecosystems. Electron. Mark. **29**(3), 503–518 (2019). https://doi.org/10.1007/s12525-019-00337-y
16. Rochet, J.-C., Tirole, J.: Platform competition in two-sided markets. J. Eur. Econ. Assoc. **1**(4), 990–1029 (2003)
17. Schreieck, M., Hakes, C., Wiesche, M., Krcmar, H.: Governing platforms in the internet of things. In: Ojala, A., Holmström Olsson, H., Werder, K. (eds.) ICSOB 2017. LNBIP, vol. 304, pp. 32–46. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-69191-6_3
18. Nickerson, R.C., Varshney, U., Muntermann, J.: A method for taxonomy development and its application in information systems. Eur. J. Inf. Syst. **22**(3), 1–24 (2013)
19. Lösser, B., Oberländer, A. M., Rau, D.: Taxonomy research in information systems: a systematic assessment. In: Proceedings of e 27th ECIS, pp. 1–17, Stockholm-Uppsala, Schweden (2019).
20. Omair, B., Alturki, A.: An improved method for taxonomy development in information systems. Int. J. Adv. Comput. Sci. Appl. **11**(4), 535–540 (2020)
21. El Sawy, O.A., Pereira, F.: VISOR: a unified framework for business modeling in the evolving digital space. In: Werder, K. (ed.) Business Modelling in the Dynamic Digital Space. SDS, pp. 21–35. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-31765-1_3

22. Hagiu, A., Jullien, B.: Why do intermediaries divert search? Rand J. Econ. **42**(2), 337–362 (2011)
23. Godes, D., Ofek, E., Sarvary, M.:. Content vs. advertising: the impact of competition on media firm strategy. Market. Sci. **28**(1), 20–35 (2009).
24. Cabral, L.: Towards a theory of platform dynamics. J. Econ. Manag. Strategy **28**(1), 60–72 (2019)
25. Guijarro, L., Vidal, J.R., Pla, V., Naldi, M.: Economic analysis of a multi-sided platform for sensor-based services in the internet of things. Sensors **19**(2), 373 (2019)
26. Chakravorti, S., Roson, R.: Platform competition in two-sided markets: the case of payment networks. Rev. Netw. Econ. **5**(1), 118–142 (2006)
27. Amelio, A., Jullien, B.: Tying and freebies in two-sided markets. Int. J. Ind. Organ. **30**(5), 436–446 (2012)
28. Hagiu, A.: Two-sided platforms: product variety and pricing structures. J. Econ. Manag. Strategy **18**(4), 1011–1043 (2009)
29. Tan, G., Zhou, J.: The effects of competition and entry in multi-sided markets. Rev. Econ. Stud. **88**, 1002–1030 (2021)
30. Schreieck, M., Hein, A., Wiesche, M., Krcmar, H.: The challenge of governing digital platform ecosystems. Digital marketplaces unleashed, 527–538 (2018).
31. Dal Bianco, V., Myllärniemi, V., Komssi, M., Raatikainen, M.: The role of platform boundary resources in software ecosystems: a case study. In: 2014 IEEE/IFIP Conference on Software Architecture, pp. 11–20, Sydney, Australia (2014).
32. Arnold, L., Jöhnk, J., Vogt, F., Urbach, N.: A taxonomy of industrial IoT platforms' architectural features. In: Proceedings of 16th International Conference on Wirtschaftsinformatik, Essen, Germany (2021)
33. Jullien, B., Pavan, A.: Information management and pricing in platform markets. Rev. Econ. Stud. **86**(4), 1666–1703 (2019)
34. Hagiu, A., Hałaburda, H.: Information and two-sided platform profits. Int. J. Ind. Organ. **34**, 25–35 (2014)
35. Evans, D.S.: Some empirical aspects of multi-sided platform industries. Rev. Netw. Econ. **2**(3), 191–209 (2003)
36. Gabszewicz, J.J., Laussel, D., Sonnac, N.: Does advertising lower the price of newspapers to consumers? A theoretical appraisal. Econ. Lett. **87**(1), 127–134 (2005)
37. Caillaud, B., Jullien, B.: Chicken & egg: competition among intermediation service providers. RAND J. Econ. 309–328 (2003)
38. Bolt, W., Tieman, A.F.: Heavily skewed pricing in two-sided markets. Int. J. Ind. Organ. **26**(5), 1250–1255 (2008)
39. Schüler, F., Petrik, D.: Objectives of platform research: a co-citation and systematic literature review analysis. In: Seiter, M., Grünert, L., Steur, A. (eds.) Management Digitaler Plattformen. Z, vol. 75/20, pp. 1–33. Springer, Wiesbaden (2021). https://doi.org/10.1007/978-3-658-31118-6_1

Assessing the Health of the Dark Web:
An Analysis of Dark Web Open Source Software Projects

Samuel Onyango, Emilie Steenvoorden, Joram Scholten, and Slinger Jansen(✉)

Department of Computer Science, Utrecht University, Utrecht, The Netherlands
{s.o.onyango,e.r.m.steenvoorden,e.j.scholten2}@students.uu.nl,
slinger.jansen@uu.nl

Abstract. A hidden part of the World Wide Web is known as the Dark Web, featuring websites that cannot be indexed by traditional search engines. Many open source software products are used to access and navigate through the Dark Web. Together they form the Dark Web open source software ecosystem. Research on this ecosystem is scarce and research on the ecosystem health is non-existent, even though ecosystem health is an useful indicator of the livelihood of an ecosystem. The goal of this research is to evaluate the health of the ecosystem through an assessment of Tor, I2P and GitHub. The Open Source Ecosystem Health Operationalization framework is used to help perform this assessment. Eight metrics from the framework are selected, which are measured using the data collected. Analysis of Tor and I2P metrics suggest that there has been an increase in Tor and I2P user activity in the recent past. Added knowledge, spin offs and forks and usage indicate active participation and interest in Tor and I2P. There has also been an increase in the number of active GitHub Dark Web projects. However, these GitHub projects are not well-connected and only a small number of projects have a large number of contributors. There is some variety among the GitHub software projects. The framework proves to be adequately capable of determining the health of the Dark Web open source ecosystem with the available data.

Keywords: Dark web · Open source ecosystem · Software ecosystem health · Open source ecosystem health operationalization framework

1 Introduction

"Whatever is done in the dark shall be brought to light" as the saying goes. The Internet is versed and layered from the Surface Web to the Deep Web and to the Dark Web. The deeper you go, the darker it becomes. The Dark Web is known for illicit activities, including the distribution of pornography, hacking, money laundering and selling guns and drugs on marketplaces [3]. However, the Dark Web is also used by non-criminal user groups, such as political activists, whistle-blowers, journalists, law enforcement agents and the military. Benefits of the Dark Web range from operating in totalitarian regimes and protecting

P. Gregory and P. Kruchten (Eds.): XP 2021 Workshops, LNBIP 426, pp. 125–134, 2021.
https://doi.org/10.1007/978-3-030-88583-0_12

sources to corporate spying and sharing confidential information [9]. In all layers of the Internet actors are involved, who communicate, have networks and use and produce software, thus making up a software ecosystem (SECO). Open source software ecosystems (OSSECOs) are defined by Franco et al. [5] as: "A SECO placed in a heterogeneous environment, whose boundary is a set of niche players and whose keystone player is an OSS community around a set of projects in an open-common platform".

The Deep Web and particularly the Dark Web are still shrouded in mystery. Therefore, more information is needed on the state of the Dark Web and the activities that keep its "engine running". A keen interest on special OSS such as The Onion Router (Tor) and the Invisible Internet Project (I2P), that are used to access the unindexed parts of the Internet, can provide insights on activity. Due to the anonymity, information on the Dark Web users and overall traffic is scarce [13]. However, even with published metrics, measuring the Internet by accurately classifying content and traffic remains challenging.

The main goal of this work is to shed some light on the health of the dark web ecosystem, by looking at the publicly available software that targets the dark web.

1.1 Related Work

The networks Tor and I2P allow anonymous access to the Dark Web. Similar anonymization software with significantly fewer users are Freenet, JonDonym, JAP and ZeroNet. Tor, developed in 2002 by the U.S. Naval Research Laboratory, is the most widely used network to access the Dark Web [13]. Tor's aim is to create a private access to an uncensored web while respecting user's privacy by connecting them to websites through virtual tunnels.

The popularity of Tor is due to the ease of use and the reliable anonymous access [10] resulting into 2 to 2.5 million users per day [8]. The I2P network is similar to Tor and facilitates anonymous website hosting by means of encryption and relays [3]. A difference is users of I2P automatically act as a "node" to transfer information, whereas Tor users must actively decide to become a node [1]. A study in 2018 showed I2P has around 32K active users on a daily basis [6]. Note that I2P can also be used to access the Surface Web. Cilleruelo et al. [4] analyzed the connection between the Tor and I2P networks by looking at the darknet services. They found Tor and I2P operate as an ecosystem and there are clear paths between them.

In 2014, Jansen [7] introduced the OSEHO framework. Contrary to other frameworks, OSEHO offers a more granular description of metrics, divided over three pillars, that can be used to establish the health of an OSSECO. The productivity pillar deals with the ability of projects to add value within an ecosystem. Robustness illustrates how well projects of an ecosystem can cope with changes that may occur in the environment. Lastly, niche creation indicates an ecosystem's ability to reinvent itself to take advantage of new opportunities. The metrics are applied at two levels. The project level consists of analyses of projects within the SECO, while the network level puts into action different elements for this ecosystem domain. The framework is created with considerations of SECO features, including licenses, code conventions, documentation, quality and public support for the projects OSEHO is applied to various SECOs already. [2] conducted research on the ecosystem health of cryptocurrencies, by focusing on the highest-valued distinct cryptocurrencies.

1.2 Research Method

The aim of this research is to assess the health of the Dark Web OSSECO by analysing two open source software projects. Tor and I2P are selected for this research because they have the largest number of users [12]. To determine the health of the ecosystem three metrics of the OSEHO framework, robustness, productivity and niche creation, are used [7]. The OSEHO framework is suitable for this study as it majorly involves analysis of quantitative data from the Dark Web OSS project sources, which correlate well with the metrics described in the framework. On top of that, the framework is designed for OSSECOs, which makes it a good fit. Quantitative data analysis is performed to measure the metrics, as well as some descriptive data analysis to further support the measuring of the metrics. The main research question of this paper is formulated as follows: *How can open source software ecosystem health be used as a proxy for measuring activity on the Dark Web?*. To answer this question a literature review on ecosystem health, the OSEHO framework and Dark Web OSS is conducted. Thereafter, quantitative data analysis is used to identify the productivity, robustness and niche creation of the Dark Web OSSECO. Data is collected from the code hosting platform GitHub, as well as the ecosystem hubs of Tor and I2P. Tor and I2P metrics identify the amount of users of the networks over the last twenty years. GitHub data includes historical data on the number of Dark Web projects and the number of developers. GitHub Dark Web project activity refers to the number of projects that are actively being updated and maintained.

2 Applying the OSEHO Framework

This section covers the application of the OSEHO framework in five steps.

Step 1 & 2 - Set Goals and Select Ecosystem Scope
The aim of this research is to evaluate the health of the Dark Web OSSECO, by conducting an analysis on GitHub Dark Web projects, as well as Tor and I2P. First the scope of the ecosystem needs to be determined. The entire Dark Web ecosystem is too broad to capture and a single OSS project is too limited to draw meaningful conclusions about the Dark Web OSSECO. Therefore, the scope of this study is limited to OSS projects in the Dark Web ecosystem retrieved from GitHub. Data is easier to find on OSS projects due to their open nature. Identification of the projects is done using the platform GitHub, which is the most widely-used hosting service for software development projects. The focus of the research is on ecosystem activity, as it is a clear indicator of health and data on activity is openly accessible. The traffic of Tor is used as activity data

source because it is the most used network to access the Dark Web. For the OSS projects the existing networks, users, forums developers and relationships under these projects are analysed.

Step 3 - Select Metrics

The OSEHO metrics are distinctively categorized in the pillars: Productivity, Robustness and Niche creation. To adhere to the scope of this study, a selection of the most relevant metrics are shown in Table 1.

Table 1. Overview of selected OSEHO metrics and sources.

OSEHO pillar	Metrics network level	Metrics project level	Source
Productivity	Added knowledge about ecosystem (1)		Collaboration forum (StackOverflow, Reddit and phpBB)
		Spin-offs and forks (2)	GitHub
		Usage (3)	Tor and I2P metrics
Robustness	Total number of active projects (4)		GitHub
	Cohesion (5)		Programmable web (API data)
		Active contributors (6)	GitHub
		Page views and search statistics (7)	Google Trends
Niche creation	Variety in projects (8)		GitHub

The selection and exclusion criteria for the metrics rely upon available data. Besides that, understandability is a consideration. Therefore, the metrics that are less complex, but yet simple and clear are selected. Thirdly, quantitative metrics are more considered for analysis and interpretation reasons, thus eliminating more quantitative metrics. The first metric *Added knowledge about ecosystem* (1) considers the rate at which new and existing information is shared within an ecosystem. On the project level, *Spin-offs and forks* (2) are relevant as it indicates the interest of developers in the projects within the ecosystem, therefore enhancing productivity and ultimately health. *Usage* (3), as a measure of health through productivity, is taken from the end user perspective. The number of users is an indicator of how well the ecosystem is doing. Moving on to *Total number of active projects* (4). The greater the number of active projects, the higher the survival chances of an ecosystem. For this metric, data on how often projects are updated are an indicator of activity. *Cohesion* (5) is how well-connected internally and externally the project network is. The "strength in numbers" [7] relates to how a good number of *active contributors/developers* (6) within an OSS project equates to health in that ecosystem. *Page views and search statistics* (7) show the popularity of an OSSECO. The *Variety in projects* (8) metric is the only niche creation metric. Assessment of project variety can determine the number of projects collectively contributing to the extension of the

OSS projects. Manifestation of a project in various technologies creates healthy niches in the ecosystem.

Step 4 - Assess and Collect Data

This section covers an assessment of data requirements and the applied data collection techniques for all metrics. *Added knowledge about the ecosystem* (1) is measured using an analysis of collaboration forums like phpBB, Reddit and StackOverflow. Indicators of added knowledge are aggregated information, blog posts and manuals. Next, data on the metrics *Spin-off and forks* (2) are collected through GitHub. Many software projects are an extension of other projects, as mentioned in the project description on GitHub. This information is used for the spin-off metric. The number of forks per software project are indicated on GitHub as well. The *Usage* (3) metric is measured on project level. Usage is measured through activity on the platform. Tor and I2P both publish anonymous user traffic. Tor provides analysis of their users, servers, traffic, performance, onion services as well as applications on the Tor Metrics website [14]. The I2P Metrics website provides historical infrastructure data from the I2P network (https://i2p-metrics.np-tokumei.net/). *Total number of active projects* (4) metric data is obtained by analysing GitHub repositories and identifying the ones that are involved in the Dark Web OSS projects. *Cohesion* (5) data is gathered from ProgrammableWeb, which is a source of API data. The activity metrics provided by Tor and I2P are both measures of all the activity on the networks, meaning this is not limited to the activity of the Dark Web only. In 2017 it was estimated by one of the founders of Tor that the Dark Web comprises only 3% of the Tor traffic. In 2020 a research estimated that around 6.7% of Tor traffic was related to the Dark Web [8].

For *Active contributors* (6) developers on GitHub are identified who contribute to different projects related to the ecosystem. *Page views and search statistics* (7) are measured using Google trends, which gives insight on how many searches are made relating to Tor and I2P. *Variety in projects* (8) which assesses the different kinds of projects that relate to the ecosystem are identified by analysis of GitHub projects.

Many OSS projects on GitHub are intended to be used on the Dark Web. The projects that are analyzed originate from several search terms, such as "Darkweb", "Dark Web", "darknet", "Tor" and "I2P". A small amount of the search results are actually related to the Dark Web and duplicate results are filtered out. A number of GitHub projects are identified because they were linked by other Dark Web OSS projects. This data is collected manually from GitHub. In total, 260 repositories are included, with information such as the project's name, last update date, number of favorites, contributors and forks as well as the type of software project. Through API endpoints from Stack Exchange, Reddit, ProgrammableWeb and GitHub, data is obtained. Tools like Postman are used to make HTTP requests.

Step 5 - Data Analysis and Results

Data on added knowledge about the ecosystem is presented in Table 2. Data from the forums StackOverflow (StO), Reddit (Rdt) and phBB (phpBB) show interest in Tor and I2P. In StackOverflow alone, thirty different topics on Tor are discussed provoking 31.783 responses. Data of I2P is mined from its own dedicated forum phpBB. A total of 810 questions lead to 1194 responses, showing the developers and users are actively involved in the projects.

Table 2. The productivity metrics data of Tor and I2P from different sources. The number of questions raised by users on different issues relates to these OSS projects on the forums and the number of corresponding responses. The usage represents the highest (high) and lowest (low) number of users actively involved during a particular period. The Tor data is collected as a total of users actively involved, while I2P is determined per daily usage, with high being the day with the highest recording of active users, while low representing the lowest recorded usage.

Productivity metrics	Tor		I2p	
Added knowledge	*(StO)*	*(RdT)*	*(phpBB)*	*(RdT)*
Number of questions	30	86	810	128
Number of responses	31.783	3085	1194	1367
Spin offs and forks	2590		803	
Usage	*Total users*		*Daily users*	
High	4.090.771		30.329	
Low	1.300.000		12.400	

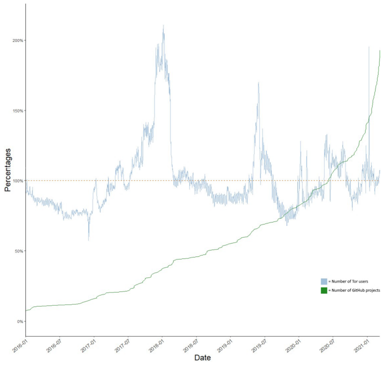


Fig. 1. The fluctuations in the number of Tor users compared to the increase of GitHub Dark Web software projects, mapped over the past five years.

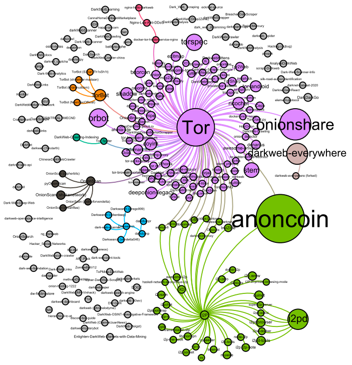


Fig. 2. Cluster visualization of the GitHub Dark Web ecosystem. The node sizes are representative for the number of active contributors a project has. Nodes with corresponding shades belong to the same cluster as assigned by the modularity algorithm.

Spin-offs and forks show many instances of projects under Tor and I2P are forked by other developers. Tor gained 2590 forks while I2P gained 803 from January 2016 to January 2021. Tor usage data shows 4.090.771 is the highest recorded number of concurrent users and 1.3 million the lowest [14]. Between 2019 and 2020 there was a sharp increase in number of users then a steady drop. Major fluctuations in these numbers are seen between 2020 and 2021 as depicted by Fig. 1. Data on I2P daily concurrent users from January 2019 till January 2021 show fluctuations with the major drops occurring between July 2019 and July 2020 with a low of 12.400 users. From November 2020 there was a steady increase to a high of over 30.000 users daily in March 2021.

With regard to robustness, roughly 260 active projects are found to be active and related to the Dark Web OSSECO. Figure 1 shows a comparison between the fluctuations in the number Tor users and active GitHub Dark Web projects. It seems like an increase in GitHub Dark Web project activity does not lead to more Tor user activity. For example, in 2021 a surge in GitHub Dark Web project activity appears, but barely any increase in Tor user activity is seen.

A Gephi network model maps the cohesion between GitHub projects in Fig. 2. It shows the projects with the largest number of active contributors, with the node size varying based on the total number of active project contributors. The model also displays the interconnectivity between projects, by looking at the edges that connect the nodes. Most of the nodes are isolated, or are connected to only one other node. The average degree of the network is 0.685, indicating a not well-connected and unhealthy network, based on the assumption that a well-connected network is healthier than networks that are not well-connected [7]. Furthermore, Tor projects appear to have far more active contributors than the I2P projects. The total number of contributors in the dataset, 1549, are not divided evenly over the projects. Out of the 260 projects that were analyzed, only 33 projects had ten or more contributors.

Google trend analysis shows Surface Web searches about Tor and I2P are significant, suggesting users are interested in the ecosystem projects Tor and I2P. From January 2016 till March 2021 major fluctuations in the number of searches are seen with the highest number being 100 and the lowest being zero.

Lastly, we analyze the variety of GitHub projects. 19 different types of software projects are identified. Most software projects are associated with crawling, networking, and security, while only a small number of projects are associated with blockchain technology, hosting, and web browsers. We conclude that there is some variety in the dark web ecosystem, but that the visible part is mostly about analyzing the dark web itself.

3 Discussion

Results show the OSSECO projects are productive, as two of the most significant contributors to health represent the productivity metric. According to [11] the productivity of OSS projects tends to be directly proportional to growth, which would validate the phenomenon observed here. Robustness is most evident through cohesion, suggesting there is some potential. Contrary to this, the number of active contributors, the page view statistics and the number of active projects weakens this pillar. Finally, niche creation is portrayed by the variety in projects. As there is some variety in the projects, we assess niche creation as moderately healthy. The results suggest an adequate amount of project variety and growth potential. Only four metrics meet the threshold required to suggest that the ecosystem is healthy, while the other four do not provide a convincing argument against a healthy ecosystem. Generally, the health of the Dark Web ecosystem is not steady at any given time. These mixed results suggest volatility, making it difficult to tell what can shift to turn the tide.

This research is subject to several validity threats. The GitHub data is collected manually, due to the choice of GitHub search terms it is possible some Dark Web OSS projects are not found. This means the used dataset could be incomplete, and should be interpreted as such. On top of that, GitHub is the only used source for OSS projects. Also, other code hosting platforms with repositories exist. The metrics that are answered using the dataset are thus only partially answered. Aside from that, Fig. 1 features Tor user data published by Tor. The metrics-timeline Git repository can be consulted to try and explain the fluctuations in the data. However, some odd peaks in the data cannot be explained by events. For example, on January 28th the user count peaked to 4.1 million, and dropped the next day to 2 million. There are no events listed in the Git repository that could possibly explain this large peak. This brings question to the legitimacy of the data. However, no good alternative for collecting data on the number of Tor users exists, due to anonymization.

3.1 Conclusion and Future Work

This paper provides an analysis of Dark Web related OSS projects determining the Dark Web OSSECO health. The OSEHO framework metrics are applied to assess the activities in this ecosystem in terms of productivity, robustness and niche creation. The study takes a unique approach by applying the framework to investigate activity on the Dark Web through the analysis of health metrics. Overall, results suggest a moderate amount of project variety and the potential to grow. All of the metrics suggest ecosystem activity but it is hard to ascertain to what percentage this represents the Dark Web as a whole. Furthermore, the health of the Dark Web ecosystem is not steady at any time.

Further research encompassing more metrics of the OSEHO framework would increase the effectiveness of the framework. The framework is implemented successfully to yield significant results showing there is activity. However, a myriad of challenges were to overcome to make this successful. First, the anonymous nature of the web made a comprehensive collection of data an uphill task leading to exclusion of some relevant metrics. Secondly, scoping the Dark Web ecosystem remains a challenge and requires future work to be done. It is ideal to have a more inclusive scope to be able to apply the framework adequately to assess Dark Web projects. Generally, the framework is practically applied to yield results that adequately answer the research question, despite the aforementioned challenges.

The OSEHO framework itself was simple to apply in practice, with many different combinations of metrics possible. Other combinations of metrics would also likely have worked, which is the real strength of the framework.

Moreover, further research is needed with the inclusion of more active OSS projects in order to measure the metrics that rely on manually collected data more accurately. This could be done by including more code hosting platforms as project sources. Additionally, more research should be done on the developers of the software project on the Dark Web to better understand who enable it.

References

1. Astolfi, F., Kroese, J. Van Oorschot, J.: I2P-the invisible internet project. Leiden University Web Technology report (2015)
2. Berkhout, M., van den Brink, F., van Zwienen, M., van Vulpen, P., Jansen, S.: Software ecosystem health of cryptocurrencies. In: Wnuk, K., Brinkkemper, S. (eds.) ICSOB 2018. LNBIP, vol. 336, pp. 27–42. Springer, Cham (2018). https://doi.org/10.1007/978-3-030-04840-2_3
3. Bradbury, D.: Unveiling the dark web. Netw. Secur. **2014**(4), 14–17 (2014)
4. Cilleruelo, C., De-Marcos, L., Junquera-Sánchez, J. Martinez-Herraiz, J.J.: Interconnection between darknets. IEEE Internet Comput. (2020)
5. Franco-Bedoya, O., Ameller, D., Costal, D., Franch, X.: Open source software ecosystems: a systematic mapping. Inf. Softw. Technol. **91**, 160–185 (2017)
6. Hoang, N.P., Kintis, P., Antonakakis, M., Polychronakis, M.: An empirical study of the I2P anonymity network and its censorship resistance. In: Proceedings of the Internet Measurement Conference 2018, pp. 379–392, October 2018
7. Jansen, S.: Measuring the health of open source software ecosystems: beyond the scope of project health. Inf. Softw. Technol. **56**(11), 1508–1519 (2014)
8. Jardine, E., Lindner, A.M., Owenson, G.: The potential harms of the Tor anonymity network cluster disproportionately in free countries. Proc. Natl. Acad. Sci. **117**(50), 31716–31721 (2020)
9. Kavallieros, D., Myttas, D., Kermitsis, E., Lissaris, E., Giataganas, G., Darra, E.: Using the dark web. In: Akhgar, B., Gercke, M., Vrochidis, S., Gibson, H. (eds.) Dark Web Investigation. Security Informatics and Law Enforcement, pp. 27–48. Springer, Cham (2021). https://doi.org/10.1007/978-3-030-55343-2_2
10. Negi, N.: Comparison of anonymous communication networks-tor, I2P, Freenet. Int. Res. J. Eng. Technol. **4**(07), 2542–2544 (2017)
11. Scholtes, I., Mavrodiev, P., Schweitzer, F.: From Aristotle to Ringelmann: a large-scale analysis of team productivity and coordination in Open Source Software projects. Empir. Softw. Eng. **21**(2), 642–683 (2015). https://doi.org/10.1007/s10664-015-9406-4

Grey literature

12. Brown, B.: 2016 state of the dark web (2017). https://www.akamai.com/it/it/multimedia/documents/state-of-the-internet/akamai-2016-state-of-the-dark-web.pdf. Accessed 31 July 2021
13. Finklea, K.: Dark web, special report for congressional research service (2015). http://www.fas.org/sgp/crs/misc/R44101.pdf. Accessed 23 Feb 2021
14. Tor: Tor Metrics (n.d.). https://metrics.torproject.org/. Accessed 22 Mar 2021

Using Guilds to Foster Internal Startups in Large Organizations: A Case Study

Tor Sporsem[1](✉), Anastasiia Tkalich[1], Nils Brede Moe[1], Marius Mikalsen[1], and Nina Rygh[2]

[1] SINTEF Digital, 7034 Trondheim, Norway
tor.sporsem@sintef.no
[2] DNV, 1361 Høvik, Norway

Abstract. Software product innovation in large organizations is fundamentally challenging because of restrained freedom and flexibility to conduct experiments. As a response, large agile companies form internal startups to initiate employ-driven innovation, inspired by Lean startup. This case study investigates how communities of practice support five internal startups in developing new software products within a large organization. We observed six communities of practice meetings, two workshops and conducted ten semi-structured interviews over the course of a year. Our findings show that a community of practice, called the Innovation guild, allowed internal startups to help each other by collectively solving problems, creating shared practices, and sharing knowledge. This study confirms that benefits documented in earlier research into CoPs also hold true in the context of software product innovation in large organizations. Henceforth, we suggest that similar innovation guilds, as described in this paper, can support large companies in the innovation race for new software products.

Keywords: Software Product Innovation · Communities of Practice (CoP) · Guilds · Employee-driven innovation · Large Organizations · Lean Startup · Maritime sector

1 Introduction

Software product innovation is challenging in large organizations because they often lack the freedom to experiment and have established routines that limit flexibility [4]. Therefore, they need to find strategies to foster innovation [10]. One way is to establish a parallel organizational structure – like an Innovation Guild – to support employees innovating. Parallel structures perform functions that the regular organization does not or is ill-suited to perform well [13]. Some examples of parallel structures include quality circles [8] and *Communities of Practices* (CoP) [18]. Although some studies indicate that such parallel structures can boost innovation [16], their role in *software product innovation* is not well-examined.

Large organizations are currently trying Lean startup approaches to give internal startups the freedom to create new software products and experiment with customers,

P. Gregory and P. Kruchten (Eds.): XP 2021 Workshops, LNBIP 426, pp. 135–144, 2021.
https://doi.org/10.1007/978-3-030-88583-0_13

much like a standalone startup [4]. Innovation frameworks such as *The corporate startup* [17], design sprints in Google [21], “FedEx Day” and “20% Time” in Atlassian [10] are increasingly gaining attention as a way of giving guidelines and standardizing innovation processes to help organizations track and support software product innovation. Simultaneously, frameworks like these do state which practices or tools internal startups should leverage to drive innovation. However, internal startups are left to explore such practices and dig up needed knowledge themselves.

To shed light on this topic and recognizing that innovation in large agile companies may be particularly challenging, we ask the following research question: *How does a large organization use CoP to support internal startups in software product innovation?* To answer, we report on a case study of software product innovation at DNV Maritime, where a CoP – based on what Spotify call “Guilds” – was successfully applied to facilitate innovation processes inspired by Lean startup.

2 Related Work

Parallel organizational structures, such as Communities of Practice (CoP), are commonly applied within software-intensive companies to support employees as problem-solving knowledge workers [13]. A community of practice is a *group of people who share a concern, a set of problems, or a passion about a topic, and who deepen their knowledge and expertise in this area by interacting on an ongoing basis* [18]. CoPs can take on different functions within an organization and evolve over time [11]. To an organization, a CoP can provide an arena for problem solving, drive strategic work, share best practice, onboard newcomers, develop professional skills, and start new lines of business [19]. To individuals, it can provide help in overcoming challenges, enabling contributing to your, improve professional reputation, provid a professional identity, and (most essential in our opinion) having fun.

In our case, DNV experimented with a CoP to support their employee-driven innovation. Without sufficient support, employee-driven innovation will fail [1]. Employees constantly need to improve their skills, share knowledge, and coordinate across the organization if they are to succeed.

Empirical studies of CoPs in software engineering are far between. Paasivaara and Lassenius [11] summarized some; [6, 7, 9]. Recently, research on the use of CoPs in Spotify (known as “guilds”) have emerged [14]. This study identified four archetypes of CoPs:

- *Book clubs* focus on "learning instead of doing", where better working methods are discussed, but decisions are rarely made.
- *Open source societies* focus on members-owned components, maintaining them, improving, and finding strategies for them.
- *Support lines* focus on onboarding, providing answers to technical issues, and facilitating solutions discussions. Core experts guide less-experienced employees.
- *Standardizing committees* align practices across the organization by creating artifacts like toolset recommendations and standards.

Communities of practice are well-researched parallel structures. However, Paasivaara and Lassenius [11] argue that researchers need to study CoPs in new contexts to understand the concept further. In this study, we answer this call and examine CoPs in the context of software product innovation.

To understand how CoPs can support organizations as a parallel structure in DNV's case, we need to present some additional literature on innovation. *Software product innovation* is defined as the creation and introduction of novel software products to the market.

Lean startup is a popular approach to software product innovation where software is developed and validated through continuous experiments with customers to minimize development costs and increase customer satisfaction [12]. It is argued that the application of the Lean startup principles (e.g. *Build-Measure-Learn* and *validated learning*) increases the speed of product development [5] and improves product-market fit [3], but also faces challenges in large organizations [15].

So, how does a large company make use of Lean startup approaches? Large organizations foster *internal startups* [4] by encouraging new corporate efforts in their own environment to enter new markets and explore new business strategies [3]. One suggested solution is *The corporate startup* [17] which offers guidelines for software product innovation in an existing organizational environment. Innovation frameworks like Lean startup and The corporate startup are based on employee participation. People pitch their ideas, and the ones with the highest potential are prioritized.

3 Case Description and Research Approach

Our case is the Maritime division of DNV, a large worldwide provider of business-to-business classification, certification, verification, risk management, training, and technical advisory services. DNV sets standards for ships and offshore structures that vessels in international waters must comply with, known as Class Rules. These rules comprise safety, reliability, and environmental requirements. DNV is operating globally and considers software products crucial for offering value to its worldwide customers. Hence, software product innovation has been part of the company's strategy to shift towards digital products and services. With 3 700 employees and headquarters in Hamburg, DNV Maritime has been using agile methods to develop software since 2008.

In 2018, the company established an innovation program based on the stage-gate innovation framework named The corporate startup [17]. Employees were invited to pitch ideas for new software products and created internal startups to develop them. These internal startups participated in a CoP, called the Innovation Guild, to support their innovative work, which is the focal point of this study.

We chose a case study [20] because we closely followed five internal startups in the between June 2020 and March 2021. We collected data in 3 different ways. First, we conducted seven interviews, asking internal startups how they work and their attitude towards guild meetings (two of them were interviewed twice). Then we did three interviews with managers on how they support the internal startups. Interviews were recorded and transcribed into 61 pages of text. Second, we collected observations from guild meetings and workshops by recording them and taking notes. Third, we used documentation on the innovation framework, such as strategic documents, status reports, and emails. Table 1 summarize our gathered data.

Table 1. Data sources

Data source	Description
Interviews	10 semi-structured interviews (7 with internal startups, 3 with managers)
Meeting notes and transcripts	Guild (CoP) meetings (6 meetings, 90 min each), workshops (2 workshops, 2 days), venture board meetings (4 meetings)
Documents	Internal documents on organization, strategies and documentation of innovation framework implementation

Data analysis was performed in three steps. First, textual data was entered into the qualitative data analysis tool NVivo. Two researchers coded the data inductively, which means that phenomenon and concepts rise from the textual data and make up themes/categories. Subsequently, we compared our categories with existing literature. We constructed codes separately followed by a comparison and discussion, ending up with a total of 150 codes. One example of a code: "Guild meetings helped me establishing contact to others with competence I needed." Further, we arranged the codes into 31 themes, e.g., "Cross functional cooperation contribute positively to internal startups" (which include the example-code above). As a last quality check, we presented our findings back to the informants. Comments were duly noted and cleared up small misconceptions.

The themes were grouped according to their impact on software product innovation. Which issue they addressed and how they supported internal startups is presented in Table 3.

4 Results

DNV created and launched new products through the Innovation framework mentioned previously to facilitate software product innovation. The framework was based on a stage-gate model described in The corporate startup [17] and guided internal startups through six stages (Table 2) from ideation (*Customer insight*) to maturity (*Sustain).* Each product idea was suggested by an employee who became an *idea owner* and responsible for their own internal startup. They had to fulfill gate criteria to proceed from one stage to another (e.g., present evidence of the customer problem or customer intent). A group of business and domain experts (*Venture board*) evaluated whether the idea owners' evidence was sufficient to fulfill the criteria and progression. Operational line managers decided what amount of worktime idea owners could take out of their original job to work on the internal startups, varying from week to week – usually between 20–100%.

Being originally operative specialists, the idea owners were unexperienced in entrepreneurship. It soon became evident that all internal startups faced common challenges and could draw on each others' knowledge to overcome them. Together with the innovation program manager, they decided to form a CoP – called *Innovation guild* – to share knowledge and find common solutions to the startups' shared problems. Besides, there was a need to establish connections to domain experts in other departments whom the internal startups had to rely on to develop their products.

Table 2. Stages of the innovation framework, criteria to proceed to next stage and key activities

#	Stage	Criteria	Key activities
1	Customer insight	Evidence of the customer problem	Conduct customer interviews
2	Viability	Evidence of the customer intent	Build and test simple prototypes
3	Proof of concept	Evidence of feasibility for building, hard evidence of the customer intent	Build and test prototypes
4	Build	Evidence of possibility for scaling	Build MVP
5	Scale	Evidence for favorable market conditions	Marketing and sales campaigns, resource planning
6	Sustain		Product improvement/sustain/retire

A mandate was made in collaboration between idea owners and managers to justify the guild's existence: "sharing experiences, solving challenges, increasing competencies, providing access to expertise and finding new ways to interact with customers." Membership was primarily open for all internal startups. In addition, line managers and stakeholders from other units were invited to participate in guild meetings (depending on the topic of interest).

The idea owners chose topics based on shared challenges they were facing at the time and what they perceived valuable to discuss together. The guild gathered biweekly, with meetings approximately 1,5 h long. Usually, the first 30 min were dedicated to idea owners sharing experiences since the last meeting, followed by the topic of interest (often presented by an invited external expert) before discussing what practices and knowledge were needed to drive innovation forward. A guild facilitator was in charge of planning the agenda and invited participants as the idea owners were far too busy handling their internal startups while juggling their departmental duties. Some weeks they worked full time on the startup, while some almost none, depending on how much their origin department allowed them.

The following subsections describe three distinct challenges that the Innovation guild was essential in solving (summarized in table 3). They are structured as a timeline, following the sequence of real-life events.

Table 3. Software product innovation challenges and achievements of the Innovation guild

Challenge	Achievements of the Guild	Impact
Idea owners lacked customer contacts and know-how to approach customers	Acquiring common practices to approach customers in exploring customer-problems	Higher quality on feedback from customers and reduced time acquiring them
Lack of guidelines on pricing digital products, need to map the existing financial expertise	Increasing expertise in pricing digital products	Obtaining a pricing solution in line with the organization's existing strategy in less time
Insufficient knowledge on building and scaling products	Improving coordination with software development unit	Managers committed to dedicating developer resources earlier

4.1 Acquiring Common Practices to Approach Customers

The first activity encouraged by the innovation framework was customer interviews (Table 2, stage 1). However, customer insight was not an established practice among the idea owners. There was no systematic way of choosing or approaching the customers, and some did not even know who to contact. One idea owner commented: "It is the gut feeling that decides which company to approach. But how do we get a systematic way to get an insight on whom to approach?" In a Guild meeting, it was deciced to involve the marketing team to find a way, and in the following marketing and sales intelligence were invited to discuss the challenge. Three participants from the marketing team presented an overview of the tools they applied to communicate with customers and analyze markets (e.g., digital marketing and sales intelligence tool, customer segment, email templates). The idea owners found the meeting helpful; one of them commented: "For me the meeting was good. The customer matrix will help me to tune my email campaign.“ In this way, the guild assisted idea owners in acquiring new practices to approach customers by leveraging the marketing experts' existing knowledge and discussing ways of using it in the startups. As a result, they were enabled to achieve higher quality customer feedback in a reduced amount of time.

4.2 Building Competence in Pricing Digital Products

According to the innovation framework, idea owners had to present evidence of customers' intent to buy the new products (table 2, stages 2 and 3). How exactly such evidence could be collected was nonetheless unclear. One proposal was to demonstrate the customers' intent by collecting their feedback on tentative pricing models. However, idea owners held no expertise in pricing. The guild initiated a series of meetings inviting representatives from finance, digital sales, and line managers to address this challenge. Finance managers realized the need to identify what possibilities the existing payment mechanisms offered concerning the new digital pricing. He expressed: "Idea owners should suggest an idea on how their products can be priced, but it is important for us to find out which pricing models we can offer for them to choose from.“ In collaboration with digital sales experts, the finance managers created a list of available pricing models with instructions on how they fit different types of offerings. One idea owner explained: “A list of what pricing models are possible and not, is great. I am really happy to see that it is happening.“ In sum, the Innovation guild supported the internal startups by finding ways of pricing digital products through acting as a collaboration arena for idea owners and pricing experts. Henceforth, they saved time obtaining a pricing solution in line with the organization’s existing pricing strategy.

4.3 Finding Ways to Collaborate with Software Developers

Entering the Build-stage (Table 2, stage 4), the prototype of an internal startup was handed to the software unit for subsequent development. However, until this stage, the idea owners had focused only on exploring the business potential. Further, most idea owners did not hold sufficient knowledge on building and scaling products. They lacked documents that software developers needed to start developing, like feature lists and user stories. One idea owner stated: “You expect to give your idea to the IT guys and then come back in two weeks or a month, and everything is ready. But this is not how it turns out to be.“ After discussions in the guild, idea owners decided to include and coordinate with the software unit earlier in the innovation process. Software developers were invited to the following guild-meeting where they described their agile practices of working with new products in other business areas of DNV, followed by a discussion among idea owners on how they can fit this way of working into their innovation process. A second guild meeting was held on the topic where one idea owner and the head of the software department had successfully collaborated. As a result, other idea owners acquired knowledge of the software development process and found inspiration on how to make this collaboration work.

The line management also acknowledged that earlier involvement of software developers should be practiced whenever possible. They committed to dedicate software developers earlier. A line manager said: “We must avoid handover to IT and build as one team.“

To summarize, the Innovation guild allowed internal startups to standardize practices to tackle shared challenges, build and share competence together, and create collaboration practices with other units. According to themselves, the Innovation guild helped internal startups to understand how to best practice innovation within DNV.

5 Discussion and Conclusions

To innovate like startups, large agile companies need to develop strategies to foster software product innovation internally [10]. However, the application of ready-available guidelines are not sufficient alone to drive internal startups to excellence. One possible solution is to engage in Communities of practice [16] or other parallel structures that improve organizational problem-solving [11, 13]. To answer our research question – *How does a large organization use CoP to support internal startups in software product innovation?* – we described how a large agile company applied a CoP to foster internal starups. In the following discussion we summarize how the CoP succeeded and compare it to the types of guilds found in Spotify.

Although innovation frameworks like The corporate startup [17] give step-by-step guidance on going from idea to product, we found that a parallel structure was needed to support it. Employees that usually carry out specialized tasks are an excellent source for new ideas. However, they typically lack experience in innovating software products. Idea owners needed to work together and draw on expertise from each other to solve problems that arose during developing collectively. The Innovation guild evolved into an arena where new practices emerged through sharing knowledge (e.g., how to develop and scale products), expand idea owners' skills (e.g., designing pricing models), improving coordination (with the software unit), and standardizing practice (such as how to work with the customers). In this way, CoPs can be seen as a prerequisite for succeeding with employee driven software product innovation that employees.

Our study's innovation guild holds similarities to communities of practice described by Wenger et al. [18], Paasivaara and Lassenius [11], and Smite et al. [14], who found that guilds support knowledge sharing, networking, and standardization. According to the archetypes identified in Spotify [14] (described in chapter 2), the innovation guild can be labeled as a *standardizing committee.* It helped idea owners create common practices and ways forward. Also, starting every guild meeting, idea owners shared their latest experiences, and guests were invited to share knowledge and inspire. This is typical for a *book club.* A mix between a book club and standardizing committee provides the startups with the best of two worlds: they can control what practices to standardize and acquire knowledge without committing to any decisions. From this, we can learn what type of guilds functions as a lubricant for innovation frameworks to work in large organizations, thus becoming a powerful organizational structure when innovating software products. Our study confirms that benefits documented in earlier research into CoPs [11, 16, 18] also hold true in the context of software product innovation in large organizations.

Despite large organizations struggling in reaching the same level of success as startups [4], they hold massive assets in terms of expertise, resources, and an established customer base. Innovation guilds activate these assets to support internal startups, hence contributing to their innovation process. In contrast, standalone startups have to establish inter-organizational alliances to access similar assets while risking their partners' opportunistic exploitation [2]. Thus, whereas startups ultimately face challenges alone, large organizations can become powerful allies for their internal startups when supported by parallel structures like an innovation guild.

Our study holds implications for innovation frameworks – and those using them – by describing how CoPs, like the Innovation guild, can supplement such frameworks in supporting internal startups. Simply implementing an innovation framework was insufficient in a large agile organization, while combining it with a parallel organizational structure like an Innovation guild drove innovation capability.

References

1. Aasen, T.M. et al.: In search of best practices for employee-driven innovation: Experiences from Norwegian work life. In: Employee-Driven Innovation. pp. 57–74 Springer (2012).
2. Baum, J.A.C., et al.: Don't go it alone: alliance network composition and startups' performance in Canadian biotechnology. Strateg. Manag. J. **21**(3), 267–294 (2000). https://doi.org/10.1002/(SICI)1097-0266(200003)21:3%3c267::AID-SMJ89%3e3.0.CO;2-8

3. Edison, H., et al.: Innovation Initiatives in Large Software Companies: A Systematic Mapping Study. Inf. Softw. Technol. **95**, 1–14 (2018). https://doi.org/10.1016/j.infsof.2017.12.007
4. Edison, H., et al.: Lean Internal Startups for Software Product Innovation in Large Companies: Enablers and Inhibitors. J. Syst. Softw. **135**, 69–87 (2018). https://doi.org/10.1016/j.jss.2017.09.034
5. Edison, H., et al.: Lean startup: why large software companies should care. In: Scientific Workshop Proceedings of the XP2015. pp. 1–7 Association for Computing Machinery, New York, NY, USA (2015). https://doi.org/10.1145/2764979.2764981.
6. Gongla, P., Rizzuto, C.R.: Evolving communities of practice: IBM Global Services experience. IBM Syst. J. **40**(4), 842–862 (2001). https://doi.org/10.1147/sj.404.0842
7. Kahkonen, T.: Agile methods for large organizations-building communities of practice. In: Agile development conference. pp. 2–10 IEEE (2004).
8. Lawler, E.E., III., Mohrman, S.A.: Quality circles: After the honeymoon. Organ. Dyn. **15**(4), 42–54 (1987)
9. Mestad, A. et al.: Building a Learning Organization: Three Phases of Communities of Practice in a Software Consulting Company. In: 2007 40th Annual Hawaii International Conference on System Sciences (HICSS'07). pp. 189a–189a (2007). https://doi.org/10.1109/HICSS.2007.115.
10. Moe, N.B., et al.: Fostering and Sustaining Innovation in a Fast Growing Agile Company. In: Dieste, O., Jedlitschka, A., Juristo, N. (eds.) PROFES 2012. LNCS, vol. 7343, pp. 160–174. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-31063-8_13
11. Paasivaara, M., Lassenius, C.: Communities of practice in a large distributed agile software development organization – Case Ericsson. Inf. Softw. Technol. **56**(12), 1556–1577 (2014). https://doi.org/10.1016/j.infsof.2014.06.008
12. Ries, E.: The lean startup: How today's entrepreneurs use continuous innovation to create radically successful businesses. Currency (2011).
13. Šmite, D., Moe, N.B., Wigander, J., Esser, H.: Corporate-level communities at Ericsson: parallel organizational structure for fostering alignment for autonomy. In: Kruchten, P., Fraser, S., Coallier, F. (eds.) XP 2019. LNBIP, vol. 355. pp. 173–188. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-19034-7_11
14. Smite, D., et al.: Spotify Guilds: How to Succeed With Knowledge Sharing in Large-Scale Agile Organizations. IEEE Softw. **36**(2), 51–57 (2019). https://doi.org/10.1109/MS.2018.2886178
15. Sporsem, T., et al.: Understanding Barriers to Internal Startups in Large Organizations: Evidence from a Globally Distributed Company. Presented at the Preprint 2021 ACM/IEEE 16th International Conference on Global Software Engineering (ICGSE) (2021).
16. Swan, J., et al.: The Construction of 'Communities of Practice' in the Management of Innovation. Manag. Learn. **33**(4), 477–496 (2002). https://doi.org/10.1177/1350507602334005
17. Viki, T. et al.: The corporate startup. How established companies can develop successful innovation ecosystems. (2017).
18. Wenger, E., et al.: Cultivating communities of practice: a guide to managing knowledge. Harvard Business School Press, Boston, Mass (2002)
19. Wenger, E., Snyder, W.: Communities of practice: The organizational frontier. Communities of Practice: The Organizational Frontier. 139–145 (2000).
20. Yin, R.K.: Applications of Case Study Research. SAGE (2011).
21. The Design Sprint — GV, http://www.gv.com/sprint, last accessed 2021/01/08.

Employee-Driven Innovation to Fuel Internal Software Startups: Preliminary Findings

Anastasiia Tkalich(✉), Nils Brede Moe, and Tor Sporsem

SINTEF Digital, 7034 Trondheim, Norway
anastasiia.tkalich@sintef.no

Abstract. To keep up with the pace of innovation, established companies are increasingly relying on internal software startups. However, succeeding with such startups is a challenging task because internal startups need to find a balance between the interests of the company and the interest of the innovator. One approach that is argued to strengthen innovation in existing companies is employee-driven innovation (EDI). This study explores this argument by examining two internal software startups in companies aligned with the principles of EDI and with a strong focus on innovation. The preliminary findings indicate that startups with EDI are characterized by commitment towards innovation, cooperative orientation, and autonomy. The findings suggest that internal software startups may be strengthened when the parent companies practice EDI.

Keywords: Internal Software Startup · Internal Startup · Software Product Innovation · Employee-Driven Innovation · Software-Intensive Business

1 Introduction

Building a successful software startup is difficult, and very few can copy the success stories of companies like Uber, Airbnb, Spotify, and Zoom. Many software startups fail before they reach their commercial potential [1]. In order to aggressively grow and scale, they need to balance high-speed innovation in extreme uncertainty with issues such as building entrepreneurial teams, acquiring a customer base, and operate in a sustainable way [2]. Innovation is also the key for established software-intensive companies. To stay competitive, they are increasingly adopting startup approaches within their own environment, thus challenging the traditional view of startups as being independent. Such internal startups [3] offer big companies flexibility and growth that can potentially boost their software product innovation. 82% of respondents in 170 large companies reported using some elements of the startup approach to accelerate their innovation [4].

However, driving internal software startups is challenging. Comparing to regular startups, the leaders of internal startups do not always have a personal stake in the final results, which makes it hard to maintain their motivation [3]. The startups may be disconnected from other crucial parts of the organizations, such as software development units, something that may hinder coordination and thus render the development process unstable [5]. Finally, being a part of a larger corporation typically reduces the innovator´s

P. Gregory and P. Kruchten (Eds.): XP 2021 Workshops, LNBIP 426, pp. 145–154, 2021.
https://doi.org/10.1007/978-3-030-88583-0_14

autonomy because they need to take into account not only external customers but also the corporate management [6]. In such a situation, the innovators may lose the freedom to pivot and experiment, which is considered crucial for all startups [3].

One approach that is argued to strengthen innovation within existing companies is known as employee-driven innovation (EDI) [7]. EDI assumes that all employees have skills and experience that can increase the organization's overall capacity to innovate under favorable conditions, which is also aligned with one of the principles of Lean Startup "Entrepreneurs are everywhere" [8]. Today´s practices in software product innovation imply that employees take an active role in driving new products [9]. One can thus expect that companies with the increased role of employees in innovation will also be more successful in internal software startups. Despite the seeming potential of EDI to strengthen internal software startups, research on this topic is only starting to emerge in software engineering. We, therefore, ask the following research question: *How do companies with employee-driven innovation (EDI) drive their internal software startups?* To answer, we are reporting preliminary findings from data collected in two Norwegian companies with EDI and with a strong focus on innovation.

2 Related work

Traditionally, startups are understood as temporary organizations focused on innovative products with little or no operative history, aiming to grow by aggressively scaling their businesses [10]. With the increasing availability of software usage and the influence of Lean Startups, software startups have become more and more widespread [11]. They rapidly find scalable business models, build new products, and pivot if necessary, which creates pressure for established software-intensive companies.

As a response, the established companies are increasingly adopting so-called internal software startups. Just as regular (external) startups, internal software startups are risk-taking and proactive initiatives that develop software under highly uncertain conditions by constantly searching for repeatable and scalable business models [11, 12]. The main difference is, however, that internal software startups are nested within the existing parent companies [6]. Unlike external startups that rely on limited resources, internal startups benefit from several existing resources (e.g., salary) [13]. Further, if the external startups are mainly driven by the needs of their external customers, the internal startups also must consider their corporate management [13]. However, various types of internal startups may differ in how much they depend on their parent company. For example, internal ventures are typically entirely supported by the resources from the parent company. In contrast, spin-offs are based on the technology from the parent company but do not entirely rely on it for the resources and eventually become independent new companies [6].

To fuel the development of new products and businesses through internal software startups, companies often rely on specific innovation strategies [14]. For example, Google relies on practices like design sprints [9], where developers are given one day to showcase a proof of concept they believe should be part of the product, whereas Atlassian applies the method "20% Time" to allow more ambitious innovation projects to be undertaken [14]. One common trait in the practices described above is that the employees' own initiative and entrepreneurial skills play a central role for the internal startups, which is characteristic of *employee-driven innovation* (EDI) [7]. EDI is seen as an informal innovation process [15], meaning that the employees are not formally assigned to innovation tasks. Therefore the EDI approach is different from traditional research and development (R&D) when novel products are developed by a dedicated formalized unit [7]. In contrast, the EDI approach allows the innovation to be driven by employees from any unit, and it is up to the employee whether to engage in the process or not.

Organizations that systematically engage in EDI seem to share a set of cultural characteristics. Based on a study of 20 companies, Aasen et al. [15] have identified nine shared features: commitment, cooperative orientation, pride, trust, tolerance, feeling of security, development orientation, openness, and autonomy. In this study, we specifically focus on the three features that were central in our data: *commitment*, *cooperative orientation,* and *autonomy* (see Table 1 for the definitions).

Table 1. Cultural characteristics of employee-driven innovation [15]

Cultural characteristics	Description
Commitment	High commitment towards innovation among employees
Cooperative orientation	Basic assumption that there is agreement to cooperate between management and employees
Autonomy	Employees have a high degree of influence in relation to the execution of various tasks

Even though it has been described how EDI is practiced in other contexts [15] and the term has been adopted by the field of Information Systems [16], it is not sufficiently studied in software engineering. Our study seeks to address this knowledge gap by examining the companies that are both aligned with the principles of EDI and focus on innovating through internal software startups.

3 Methods

To answer the RQ, we collected data from two cases of internal software startups in two Norwegian companies (Table 2). We selected companies with internal software startups and where employees were first involved in the innovation process by taking an informal innovator role on their own initiative (EDI). Both companies are technology-intensive with a strong focus on innovation. The first company is Iterate that was listed among the 100 best workplaces for innovators in 2020 [17]. The second company is MarComp (real name suppressed for anonymity), which has since 2018 been actively training personnel and management in software product innovation. In both companies, we conducted semi-structured interviews and collected documents and meeting notes between September 2020 and April 2021. Different sets of questions were used to interview product managers comparing to interviewing other stakeholders (e.g., CEO, innovation facilitators). Examples of the questions for product managers (employees in charge of the startups) are "*Please, tell us about your innovation project and how it arose*," "*Who has been the most important collaboration partner*?", "*What has been the biggest challenge, and how did you solve it?*". The interview guide for the other stakeholders consisted of questions like "*How does your company captures ideas for new internal startups?*", "*How do you decide which ideas are good enough?*". In terms of the documents, we collected slides, emails, and websites that reflected the nature of the startups, their history, and the organizational context they operated in.

Table 2. Data sources

Parent firm	Industry	Startup	Data sources	#	Details
Iterate (Norway)	Venture-builder/IT	CalcTool	Interviews	4	Product manager, CEO, Designer, Customer
			Meeting notes	2	Informal meetings with CEO
			Documents		Website, product strategies (slides), emails
MarComp	Maritime	ShipDash	Interviews	3	Product manager (two times), innovation facilitator
			Meeting notes	5	Community of practice
			Documents		Innovation program's slides, emails

For the current analysis, we first created an overview of the cases based on the contextual characteristics that could be compared across the cases. Then we identified the data instances that had to with how the parent companies drove the startups. Finally, we grouped the innovation strategies thematically in an attempt to categorize different types of strategies. The data were analyzed by the first author, who continuously consulted the other authors to validate the emerging results. The earlier version of the paper was also shared with the key interviewees to collect their feedback, which was later incorporated in the final version.

4 Results

We will describe the internal software startups and how the parent companies drove them (innovation strategies). Key contextual characteristics of the case startups are presented in Table 3.

Table 3. Overview of the internal software startups

Contextual characteristics	CalcTool	ShipDash
Size of the parent company	S	L
Beginning of the startup job	Q3,2020	Q4 2019
Type of internal startup	Spin-off	Internal venture
Dedicated employee (product manager)	✔	✔
The startup job is financed by the parent company	Partially	✔
The product manager validates strategic decisions with the parent company		✔
Size of the startup team	5	6
Innovation coaching	✔	✔

Note. ✔ = the characteristic is identified in the startup

CalcTool. The aim of the startup is to create a new software-enabled online tool for construction engineers. Iterate (the parent company) functions as a venture builder investing in software startups and additionally provides IT consultancy services. Apart from the product manager, the CalcTool team consists of two software developers and two designers from the same company who volunteered to join the initiative. Interested in the idea, the team members agreed to use their own hour-budget sponsored by Iterate ("Iterate time") to work on the startup. The product manager expressed: "*I tried to recruit others and convince them to donate their Iterate time to me.*" The team closely collaborates with potential users (construction engineers) who provide input on the desired functionality. The product manager emphasized: "*It is very beneficial to be a part of the construction engineer network.*" The team's intention is to create an independent spin-off where Iterate can potentially become an investor. The team has full autonomy about the technical and strategic decisions with no influence from Iterate. The startup's product manager is a full-time employee at the parent company and works with the startup only 10 hours per month in addition to his spare time. He commented: "*I am willing to invest my time in this because I can get a share in the company that can come out.*" Although Iterate is only partially financing the startup job through "Iterate time," this job is very encouraged by the CEO. The startup receives support from Iterate in the form of:

- coaching - whenever the product manager requests it, the top manager acts as a coach. This is a way for the managers to be involved in the startup without taking too much control. The CEO commented: "*If people feel that they have to validate things with us […] it will go too slow. But when there is something to show, we are glad to be in a dialog*";
- networks – there is an external Slack channel on product management that is actively used by many other employees inside and outside the company and is part of the informal project management school driven by the executive manager. Earlier, the product manager worked in another startup team in the parent company, which gave

him experience in the startup work. He emphasized the importance of collaborating with other employees: "*It gives me confidence in being as I am.*";

- ceremonies - examples of the ceremonies are 1) breakfast meetings (employees meet from 7.30 to 8.30 on certain days for pitching their entrepreneur ideas); 2) "While we wait" conference on Wednesdays (11-12) where most experienced employees present their current startups; 3) "Ship-it day" – A 24-hour hackathon that can (once per year) be used for delivery in one's own project;
- culture – innovation mindset is part of the organizational culture. Individual employees are encouraged to create their own software startups and collaborate with each other to spur creativity, experience, and motivation. A designer from another project expressed: "*In Iterate, we have a shared culture that helps us collaborate even if we never worked with each other before.*"

ShipDash is developing an algorithm-enabled tool for insight into ships' emissions based on various data sources (e.g., AIS, reported fuel data from vessels). The startup is a part of an internal innovation program, and in November 2020, moved to the operational department for scaling. Since the startup is entirely supported by the resources from the parent company, it can be categorized as internal venture [6]. Key strategic decisions (e.g., scaling, extra financing, brand name) are validated with the parent company, which can slow down the development process. The product manager confessed: "*It is so challenging sometimes, because there are so many [managers] who have an opinion on what we are building should be.*" However, the innovation facilitators are constantly working on expanding the product manager´s autonomy in the firm, thus enabling faster user testing and pivoting. For example, sub-branding was introduced to lower the threshold for the product managers to test their solutions with the existing customers without harming the main company brand. The product manager of ShipDash said: "*Sub-branding is cool; it makes me want to innovate more [...]. Now I can have a label [towards customers] that I am just testing this and learning*". Since August 2020 (building and scaling), the product manager is working on the startup most of her time (85% on average), but in the past, the workload varied from 20% to 50%). This became possible because her line manager accepted her working less on the main position in favor of the startup. The parent company supports the startup by:

- coaching – provided by both in-house and external innovation coaches and internal mentors (experienced leaders, domain experts, innovation facilitators); The coaches and innovation facilitators also function as problem-solvers when the product managers are stuck. One product manager commented during a meeting: "*If we said, oh we need to have more customer contacts, then the coach said Ok, I will ask this person, and he will organize something";*
- networks – 1) customer relations managers gave access to test-users during the experimentation phase; 2) stakeholders from marketing, customer relations, finance, operations, and software were frequently invited to take part in strategic and technical decisions on both periodic (ceremonies) and non-periodic arenas (meeting requests): 3) informal networks with other product managers who shared their experience from similar startups were also crucial. The product manager emphasized: "*I relied so much*

on my good colleagues who have been through similar projects. […] I asked one of them so many questions that he is now calling himself my mentor";

- ceremonies – 1) Periodic "innovation board" meetings where domain experts evaluate the startup's progress; 2) "innovation guild" – meetings of a community of practice driven by other product managers and innovation coaches to share experiences and drive the internal startups;
- processes - using a stage-gate innovation process based on the Lean Startup approach, the company encourages the employees to suggest ideas (Call for ideas) and then develop a minimal viable product with high potential for scalability. The innovation process also implies that the product managers have the freedom to conduct user testing with the customers as part of the product development.

5 Discussion and Practical Implications

To answer the research question *How do companies with employee-driven innovation (EDI) drive their internal software startups?* we have presented preliminary findings from two cases of startups in companies with EDI. In both cases, the employees were working with the startups only part-time, but the workload seemed to vary and generally increased along with the startup's maturity. Both parent companies were relying on a set of innovation strategies to motivate their employees to engage in internal startups, such as coaching, networks, and ceremonies. Whereas the larger company had a stronger focus on how to structure the innovation processes, the smaller company focused more on shared innovation culture. We will now discuss our findings against the cultural features of EDI and earlier research on internal startups to understand the differences between internal startups with and without EDI. Due to the limited format of the workshop paper, we focus only on the three cultural features of EDI that were particularly central in our findings: *commitment*, *cooperative orientation,* and *autonomy* [15].

Our case startups were characterized by a strong commitment to the innovation process and outcome. The product manager of CalcTool was willing to work on the startup in his spare time. The startup team contained other employees that were committed to due to the shared innovation mindset in the company. In ShipDash the line managers showed their commitment to the innovation process by reducing the product manager workload, which freed them for the startup work. As a result, the product manager was willing to take up a new role in the company, as the startup was gradually occupying most of her work time. Earlier research on internal startups demonstrated that maintaining the motivation of employees to innovate is limiting for internal startups [3]. Our findings indicate that this problem can be solved by EDI that creates conditions where both employees and managers are motivated to contribute to internal software startups.

Our results suggest that companies with EDI are actively promoting cooperation not only between employees and management (as described earlier [15]) but also between employees of different departments and externally with customers. Iterate was using a plethora of ceremonies to promote networking among the employees where they could pitch their ideas, receive feedback from more experienced innovators and recruit others to the startup team. In ShipDash the parent company was also promoting the product managers´ networks with key customers, domain experts (marketing, finance, software), and more experienced product managers. Earlier research indicated that access to existing networks of experts internally in the company is enabling for the internal startups [3]. External networks with customers or corporate partners are also important for startups in general because they tend to depend on the innovation ecosystem around [2]. Based on our findings, we can thus conclude that organizations with EDI can have a positive effect on internal startups by promoting networks both internally and externally.

Finally, our findings indicate that achieving autonomy is central for internal software startups in EDI-oriented companies but that it can also be challenged. Specifically, CalcTool enjoyed high autonomy and did not have to validate the strategic decisions with the management, whereas in ShipDash the product manager´s autonomy was lower. However, MarComp was systematically working to increase the autonomy of the innovators (e.g., freedom to experiment, introduction of sub-branding). Autonomy in the decision-making process is crucial for internal startups because it speeds up development and learning through experimenting [3]. Even though EDI-companies can differ in the degree of the startups´ autonomy (due to differences in the companies´ size, financial or legal concerns), such companies still acknowledge the importance of it and actively work to increase it by different means.

6 Conclusions, Limitations, and Future Work

To gain more knowledge on internal software startups in companies with employee-driven innovation (EDI), we have reported preliminary findings from two startups in different companies. We found that both startups were characterized by high commitment towards innovation, cooperative orientation, and autonomy, which we argued is positive for internal software startups in general. One of the central limitations of this preliminary study is that the data originates from Norwegian companies where the tradition of employee involvement is historically strong [15]. In future work, we intend to collect data from other countries and conduct an even more rigorous analysis of a larger number of startups to acquire additional insight into the effect of EDI.

Acknowledgments. This research was supported by the 10xTeams project and the Research Council of Norway through grant 309344.

References

1. Crowne, M.: Why software product startups fail and what to do about it. Evolution of software product development in startup companies. In: IEEE International Engineering Management Conference. pp. 338–343. IEEE (2002).
2. Abrahamsson, P., Bosch, J., Brinkkemper, S., Mädche, A.: Software Business, Platforms, and Ecosystems: Fundamentals of Software Production Research (Dagstuhl Seminar 18182). Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik GmbH, Wadern/Saarbruecken, Germany (2018).

3. Edison, H., Smørsgård, N.M., Wang, X., Abrahamsson, P.: Lean Internal Startups for Software Product Innovation in Large Companies: Enablers and Inhibitors. J. Syst. Softw. **135**, 69–87 (2018). https://doi.org/10.1016/j.jss.2017.09.034
4. Kirsner, S.: The Barriers Big Companies Face When They Try to Act Like Lean Startups, https://hbr.org/2016/08/the-barriers-big-companies-face-when-they-try-to-act-like-lean-startups, (2016).
5. Sporsem, T., Tkalich, A., Moe, N.B., Mikalsen, M.: Understanding Barriers to Internal Startups in Large Organizations: Evidence from a Globally Distributed Company. In: Proceedings of 16th ACM/IEEE International Conference on Global Software Engineering (ICGSE). , Virtual conference (2021).
6. Edison, H., Wang, X., Jabangwe, R., Abrahamsson, P.: Innovation Initiatives in Large Software Companies: A Systematic Mapping Study. Inf. Softw. Technol. **95**, 1–14 (2018). https://doi.org/10.1016/j.infsof.2017.12.007
7. Høyrup, S.: Employee-driven innovation and workplace learning: basic concepts, approaches and themes. Transf. Eur. Rev. Labour Res. **16**, 143–154 (2010). https://doi.org/10.1177/1024258910364102
8. Ries, E.: The lean startup: How today's entrepreneurs use continuous innovation to create radically successful businesses. Currency (2011).
9. The Design Sprint — GV, http://www.gv.com/sprint, last accessed 2021/01/08.
10. Giardino, C., Wang, X., Abrahamsson, P.: Why Early-Stage Software Startups Fail: A Behavioral Framework. In: Lassenius, Casper, Smolander, Kari (eds.) ICSOB 2014. LNBIP, vol. 182, pp. 27–41. Springer, Cham (2014). https://doi.org/10.1007/978-3-319-08738-2_3
11. Paternoster, N., Giardino, C., Unterkalmsteiner, M., Gorschek, T., Abrahamsson, P.: Software development in startup companies: A systematic mapping study. Inf. Softw. Technol. **56**, 1200–1218 (2014). https://doi.org/10.1016/j.infsof.2014.04.014
12. Melegati, J., Guerra, E., Wang, X.: Understanding Hypotheses Engineering in Software Startups through a Gray Literature Review. Inf. Softw. Technol. **133**,(2021). https://doi.org/10.1016/j.infsof.2020.106465
13. Edison, H., Wang, X., Abrahamsson, P.: Lean startup: why large software companies should care. In: Scientific Workshop Proceedings of the XP2015. pp. 1–7. Association for Computing Machinery, New York, NY, USA (2015). https://doi.org/10.1145/2764979.2764981.
14. Moe, N., et al.: Fostering and Sustaining Innovation in a Fast Growing Agile Company. In: Dieste, Oscar, Jedlitschka, Andreas, Juristo, Natalia (eds.) PROFES 2012. LNCS, vol. 7343, pp. 160–174. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-31063-8_13
15. Aasen, T.M., Amundsen, O., Gressgård, L.J., Hansen, K.: Employee-driven innovation in practice - Promoting learning and collaborative innovation by tapping into diverse knowledge resources. LifeLong Learn. Eur. 4, (2012).
16. Opland, L., Jaccheri, L., Pappas, I., Engesmo, J.: Utilising the innovation potential - a systematic literature review on employee-driven digital innovation. In: 29th European Conference on Information Systems (2021).
17. Fast Company: Best Workplaces for Innovators 2020, https://www.fastcompany.com/90527870/best-workplaces-for-innovators-2020, last accessed 2021/07/08.

Towards a Framework to Guide the Creation of Development Practices for Software Startups

Jorge Melegati(✉)

Faculty of Computer Science, Free University of Bozen-Bolzano, Bolzano, Italy
jorge@jmelegati.com

Abstract. The research on software startups has increased lately, focusing on describing how these companies' unique context influences development practices. The next step for research is the creation of specific practices for these companies grounded in scientific results. An obstacle in this path is which dependent variable these novel practices should improve. A natural answer is these companies' success. This position paper reviews the literature on new ventures and startups' success to show that telling if a startup is successful or not is a complex issue. As a solution to this problem, this paper proposes a conceptual framework, suggesting that novel practices should improve success determinants or reduce inhibitors rather than focusing on the startups' success. Three examples illustrate the framework's use: hypotheses engineering, microservices, and BizDev. The identification of contributors and inhibitors for success of software startups could enrich the framework and indicate possible avenues for the creation of development practices specific tailored for these companies.

Keywords: Software startups · Startup success · Software success · Software engineering practices

1 Introduction

Recently, the research in software startups has increased in number and rigor [4], focusing on describing the consequences of this unique context to software engineering [15]. For instance, Giardino et al. [11] proposed the Greenfield Startup model to describe software development in startups as a sped up process leading to the consequence of accumulated technical debt. However, the creation of specific development practices tailored to the unique context these companies face is still in its infancy [15]. A natural question towards this goal is what the dependent variable is, i.e., which aspects these novel practices should improve to be considered useful. Besides guiding the creation of novel practices, a better definition of what is a successful startup allows the investigation of what makes some startups achieve better results allowing others teams to replicate practices employed.

P. Gregory and P. Kruchten (Eds.): XP 2021 Workshops, LNBIP 426, pp. 155–164, 2021.
https://doi.org/10.1007/978-3-030-88583-0_15

Software startups are companies searching for a scalable business model for a new software-intensive product [19]. Hence, a logical answer to the dependent variable question is success. However, this answer leads to at least two other questions: what is success for a software startup and to which extent it could be influenced by software development.

This paper explores possible answers to these two questions using results from the literature and examples of failed startups. Based on a review of the literature about success of new ventures and startups, and its determinants, I argue the complexity to define and to assess what a successful startup is, but acknowledge the existence of several contributors and inhibitors to success. Thus, I propose a conceptual framework to guide the creation and evaluation of specific software development practices to startups. According to the framework, instead of focusing on success, studies should aim to improve a contributor or reduce an inhibitor without negatively contributing to other aspects. To illustrate the framework use, I employ it to propose the evaluation of practices that, although not specifically proposed for software startups, could be employed in this context: hypotheses engineering, microservices, and BizDev.

2 Success of Software Projects

A logical approach to guide the creation of development practices for software startups is to distinguish the software project, or "technical," success from the business outcomes. However, this dichotomy is contrary to the literature on software success. The discussion of software project success and failure is a complex argument and to present it in detail would not fit this paper. Nevertheless, an overview, presented below, is essential to the flow of argument.

Ralph and Kelly [17] investigate the success concept in the software engineering context. First, they observe how the understanding of this concept evolved in project management literature. They acknowledge that, for a long time, a model used to describe such a phenomenon was the Iron, or Project, Triangle [3]. This metaphor postulates that the quality of a product (inner of the triangle) is constrained by three vertices: scope, budget, and schedule [1]. Then, the authors describe a more comprehensive taxonomy proposed by Shenhar et al. [18], which characterizes success through four dimensions: project efficiency, impact on the customer, business success, and preparing for the future. Finally, they interviewed 191 design professionals, where 68 were in the software industry, to investigate the practitioners' perceptions on success. The results indicated that professionals understand software engineering success "as a multidimensional variable comprising project efficiency, artifact quality, market performance, and stakeholder impacts over time."

Therefore, software project success does not concern only the quality, scope, and schedule of the delivered software system. Following Shenhar et al.'s taxonomy, other measures that seem especially important for software startups, are, for instance, fulfilling customer needs, commercial success, and creating a large market share. As Ralph and Kelly [17] observe, practitioners perceive the existence of a client who has problems to solve or needs to be fulfilled. Software startups face an even harder challenge: they must find not only the problems but even the customers. Some types of pivots, strategical changes on a startup's business model or product [2], support such an argument. For instance, a customer segment pivot is a shift from one customer segment to another [2]. Zoom-in pivots occur when a single feature becomes the whole product and zoom-out pivots, when a product becomes only a feature of a bigger solution.

In summary, the success of a software system goes beyond developing a set of features in a determined period of time using a prescribed budget. It also means fulfilling customer needs and reaching commercial success. Therefore, in the context of software startups is natural to expect the company success. The next section presents a review on this aspect.

3 Success of Startups

Some studies in the literature focused on the success of startups, either to learn from them, e.g. [21], or to predict them, e.g., [7]. To do so, researchers had to define what a successful startup is. Zaheer et al. [21] considered companies that had achieved one or more of the following: continued survival, a high sales volume, a stock exchange listing, or acquisition. For Dellermann et al. [7], it was sufficient to have received a series A funding, that is, "a venture capital backed funding that allows angel investors to exit the startup." A quick search on specialized media shows that these criteria for success are spread among practitioners.

However, these aspects are problematic in two ways. First, it is hard to precisely assess them either by the time needed or by the levels distinguishing different outcomes. For instance, how long should a startup run to be considered successful? Or what is a high volume of sales? Second, and most important, there are several companies that featured these criteria but were considered failures at the moment of assessment or later. A criterion based on stock exchange listing does not work, for instance, when analyzing startups that failed during the dot-com bubble. Another issue with adopting this rule is a recent trend to stay private rather than becoming public [9]. Startups can decide to not run an Initial Public Offering (IPO) to, among other reasons, wait for a better moment or to avoid the compliance required as a public company. Regarding the acquisition criterion, a counterexample is the concept of acquihire, defined by the Cambridge Dictionary[1] as: "to acquire (= buy) a company in order to use its employees' skills or knowledge, rather than for its products or services." Thus, a startup whose product failed might still be acquired by a larger company as a way to hire all team members at once. With respect to investments received, a piece of evidence against the criterion comes from a study by Cantamessa et al. [5], that investigated the failure of 214 startups based on postmortems. Among these companies, 14% failed after more than five years operating and, in this cohort, startups closed an average of 2.16 round of investments, receiving, in average, 16.39 million dollars.

[1] https://dictionary.cambridge.org/dictionary/english/acquihire.

It is beyond the scope of this paper to thoroughly define what a successful startup is. However, this brief discussion showed the complexity of the success concept for startups. A reliable assessment of this aspect requires a thorough, holistic inspection of the company, combining several aspects of the business and the product, to overcome the deficiency of employing a single criterion [12]. Besides the difficulty to reliably judge which startups are successful, there is an even major concern to software engineering research in this context: to which extent development techniques could influence the success of these companies.

Some studies focused on identifying determinants of success in high-tech new ventures. Based on a survey with 27 venture capitalists, Kakati [12] concludes that the critical determinants are entrepreneur quality, resource-based capability, and competitive strategy. Besides that, an interesting result is the distinction between winning and qualifying criteria. Winning aspects are those "which directly and significantly contribute to wining business," while qualifying criteria may not be "major determinants of success," but "any reduction [...] will be particularly serious if it drops below the critical level." The authors also performed a principal component analysis to identify factors that could explain the variance from success and failure, finding nine factors. The first factor, "incapability risk," is related to the capabilities needed to succeed. The second, "inexperience risk," captures the lack of track record and market knowledge. The third, "product risk," is the first that could be associated to software development. In the case of novel ideas, as the case of software startups, the authors acknowledge that "there is a risk of whether the product can be produced and commercialized" and even "technically elegant products may fail to exploit the untapped market."

A similar result comes from the investigation of startups' failures. These studies focus on failures rather than success since, by making these issues evident, their mitigation is easier, and more startups could be successful. In their investigation of startup failures, Cantamessa et al. [5] observed a "typical failure pattern related to the Business Development process." The authors analyzed 214 postmortems and identified one or more reasons to failure. The first two aspects were wrong business model and the lack of business development, occurring in 35% and 28% of the cases, respectively. Behind these issues, the authors observe "a high focus on the product or service by the management and founders, but an insufficient attention to commercial development." After the third position being running out of cash, happening in 21% of the cases, the fourth aspect is lack of product/market fit, in 18% of the cases, another issue related to the business development.

A natural reaction to this conclusion is that founders should focus on improving the business development process and the software engineering team should care after the business elements are set. However, experimentation, the basis of the business development usually advocated for startups, where product assumptions are taken as hypotheses and tested, represents a unique problem for software engineering, since currently available methods for software development revolve around requirements obtained with varying degrees of customer contact frequency [15]. This uniqueness means several consequences to software development. First, since experiments are based on hypotheses and not in requirements as conventional practices, requirements engineering practices might not be the most suitable to the this context. Second, the heavy use of experiments and the probable consequence of the implemented features influence the software system's architecture and design. Bad choices could contribute to technical debt accumulation and compromise the system capacity to handle larger loads or the team ability to maintain the code or add new features during scaling the product. Third, startups are characterized by small teams where members often perform multiple and diverse tasks. A software developer in a startup has a higher probability to be involved in other tasks than programming.

In summary, defining whether a startup is successful or not is a complex endeavor and represents a challenge to studies aiming to either learn from these companies or to create novel techniques to improve these companies' chance of success. A more viable option is to focus on improving (or detecting the existence of) determinants of success and reducing aspects that increase the risk of failure. To summarize this idea, the next section proposes a conceptual framework to guide the research on practices specific for software startups.

4 Conceptual Framework

Based on the review above, it is complex and, often, dubious to assess if a startup is successful or not. This section presents a conceptual framework to support researchers, or practitioners, interested in developing, or evaluating, practices to software startups. It is essential, though, to highlight that this framework does not aim to answer what success is for these companies.

Research has identified many contributors and inhibitors to success, that is, aspects influencing the success of a startup either positively (contributors) or negatively (inhibitors). Similarly, several other aspects, including software engineering practices, could influence positively (or negatively) these antecedents of success. Clearly, not all of them could be influenced by software development practices. For instance, changes on legal or economical frameworks might derail a startup's business model. To simplify the framework, I did not explicitly model aspects that could act as a contributor or inhibitor depending on the level. These aspects could be modeled in a way to fit this framework without losing generality.

Figure 1 depicts the framework using arrows to represent the influence of several aspects, represented by boxes, including software development practices to guide the creation of practices specific to software startups. The labels associated with the arrows tells if the influence is positive (represented by the '+' arrow), negatively ('−' arrow), or not ('0' arrow).

To use the framework, a researcher, aiming to create a novel technique, should identify which aspects associated, either positively or negatively, to startup success that activity could influence. Then, the researcher should demonstrate that the new technique improves contributors, reduce inhibitors, or both. The proposed technique should also lead to no, or limited, negative consequences or the

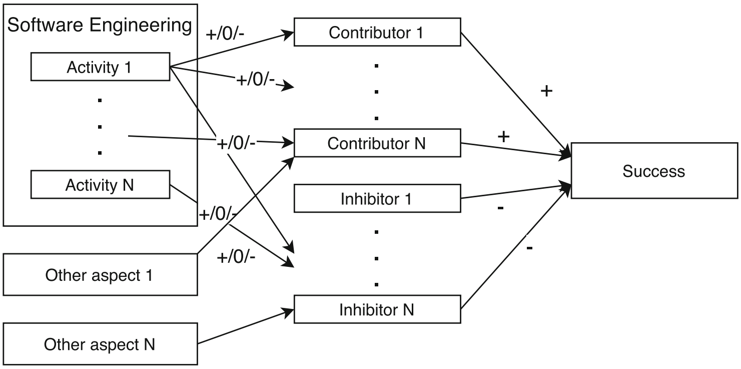


Fig. 1. A conceptual framework to guide the creation of practices for software startups.

researcher should, at least, acknowledge this issue and point out this problem as an issue to be solved. In this initial version of the framework, I do not list possible contributors and inhibitors to success but, in future work, this addition could be achieved by reviewing the literature on new ventures and inspecting failure cases. This list creation process should also consider the different stages a startup faces [13]: inception, stabilization, growth, and maturity. Some contributors and inhibitors could be associated with goals of specific stages. For instance, in the stabilization stage, the startup should prepare for scaling in the next stage. If a technique is proposed to be used in this stage, it should not have negative consequences to this goal.

5 Some Examples of the Framework Use

To exemplify the framework use, this section describes how researchers could evaluate if three proposed practices from different areas of software engineering are useful for software startups based on their influence on a contributor and on an inhibitor to success. Based on the review presented in Sect. 3, the business development speed is a contributor, an aspect positively associated to startup success. On the other hand, the accumulated technical debt could act as an inhibitor, that is, negatively associated, to success, especially when a startup reaches the scaling stage.

Hypotheses Engineering [16] is a proposal to support experiment-driven software development, as a parallel to Requirements Engineering in the conventional, requirements-driven software development. Rather than describing features to be implemented, hypotheses define uncertain questions about the business model or other software product aspects that the team could assess with an experiment. Since Requirements Engineering's goal is to precisely define the problem the software should solve [6], software startups represent a unique challenge to development. A customer on whom to rely to elicit requirements, a feature present in traditional and agile methodologies alike [15], lacks in the context of software startups. Therefore, Hypotheses Engineering could be a valuable practices for software startups.

Following the framework, an evaluation of the Hypotheses Engineering for software startups could assess, for instance, the speed with which startups adjust their plans based on information about the customer and market rather than the companies' success.

However, although the use of experiments can reduce technical debt by not building unnecessary features, it could also increase the technical debt [20]. For instance, one type of experiment is a prototype to deliver features as soon as possible to get customers' feedback. However, these quick solutions probably lead to high levels of technical debt. Therefore, an evaluation of Hypotheses Engineering for software startups should also assess the influence of these practices to technical debt.

Microservices is "an approach to developing a single application as a suite of small services, each running in its own process and communicating with lightweight mechanisms" [14]. It could be useful for a fast changing product as a way to isolate features, remove them if not useful, facilitate the software modularization, and scale. Using the framework, rather than assessing success, the employment of microservices should facilitate business experimentation while keeping technical debt low.

BizDev is a role proposed by Fitzgerald and Stol [10], as a closer integration between business and software development functions, a parallel to the well-known DevOps phenomenon. DevOps is "an organizational shift in which [...] cross-functional teams work on continuous operational feature delivery" by employing "automated development, deployment, and infrastructure monitoring" [8]. That is, instead of separated teams, development and operations work together in the delivery of features. Similarly, BizDev proposes collaboration between development and business strategy [10]. This new role could be an answer for the biggest issue for professionals in software startups: the mindset shift from strictly developing a feature defined by customers or other stakeholders to an active participant in the business development. For instance, several pivots mean partial or total modifications to software [2]. To evaluate the effectiveness of this new role, a researcher could evaluate if it indeed increases the speed on which the startup performs business development.

Figure 2 summarizes how the framework could be used to evaluate these three proposals when applied in software startups. Rather than measuring the companies' success, using or not these techniques, researchers could test, for instance, if they improve the speed of business development without increasing the technical debt.

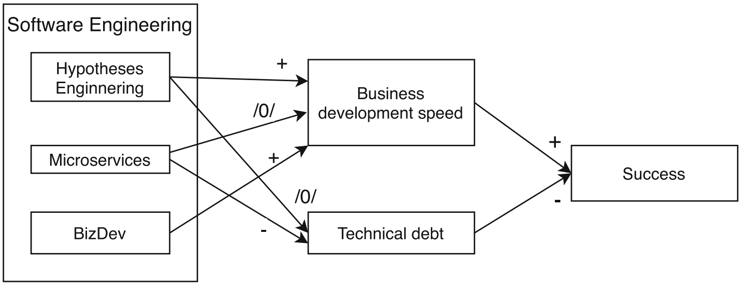


Fig. 2. Three examples applying the framework.

6 Conclusions

Researchers focused on improving software engineering practices for startups depend on clearly defining which dependent variable to improve. Although success is a natural answer, to tell if a software startup is successful or not is a complex task, requiring an holistic analysis of the company. This issue also hinders research aiming to learning from successful cases. To support these researchers, this paper reviews the literature to argue that success is complex and studies should focus on improving determinants and reducing inhibitors of success. To operationalize this argument, I propose a conceptual framework and, to illustrate its use, I gave three examples of how it could help the evaluation of novel practices from diverse areas of software engineering in startups.

The argument presented in this paper is not essentially new and a similar one has been implicitly employed in software engineering research. That is, researchers evaluate proposed techniques against diverse aspects rather than software project success. However, the uniqueness of software startups induces the use of success as a final goal, or a reference, to the development of new practices. This paper argued that this choice is not practical. Besides that, it showed that focusing on diverse aspects than those generally used in software engineering research might be useful as the case of business development. This paper is an initial proposal and, in future work, we will develop the framework further identifying contributing factors from the literature and inhibitors from failed startup cases which are abundantly available in Internet.

References

1. Atkinson, R.: Project management: cost, time and quality, two best guesses and a phenomenon, its time to accept other success criteria. Int. J. Project Manag. **17**(6), 337–342 (1999)
2. Bajwa, S.S., Wang, X., Nguyen Duc, A., Abrahamsson, P.: "Failures" to be celebrated: an analysis of major pivots of software startups. Empirical Softw. Eng. **22**(5), 2373–2408 (2017)

3. Bano, M., Zowghi, D., da Rimini, F.: User satisfaction and system success: an empirical exploration of user involvement in software development. Empirical Softw. Eng. **22**(5), 2339–2372 (2017)
4. 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)
5. Cantamessa, M., Gatteschi, V., Perboli, G., Rosano, M.: Startups' roads to failure. Sustainability **10**(7), 2346 (2018)
6. Cheng, B.H.C., Atlee, J.M., Joanne, M.: Research directions in requirements engineering. In: Proceedings of the 2007 Future of Software Engineering, FOSE 2007, pp. 285–303 (2007)
7. Dellermann, D., Ebel, P., Lipusch, N., Popp, K.M., Leimeister, J.M.: Finding the unicorn: predicting early stage startup success through a hybrid intelligence method. In: ICIS 2017: Transforming Society with Digital Innovation (2018)
8. Ebert, C., Gallardo, G., Hernantes, J., Serrano, N.: DevOps. IEEE Softw. **33**(3), 94–100 (2016). https://doi.org/10.1109/MS.2016.68
9. Ewens, M., Farre-Mensa, J.: The Evolution of the Private Equity Market and the Decline in IPOs (2017)
10. Fitzgerald, B., Stol, K.J.: Continuous software engineering: a roadmap and agenda. J. Syst. Softw. **123**, 176–189 (2017)
11. Giardino, C., Paternoster, N., Unterkalmsteiner, M., Gorschek, T., Abrahamsson, P.: Software development in startup companies: the greenfield startup model. IEEE Trans. Softw. Eng. **42**(6), 585–604 (2016)
12. Kakati, M.: Success criteria in high-tech new ventures. Technovation **23**(5), 447–457 (2003)
13. Klotins, E., et al.: A progression model of software engineering goals, challenges, and practices in start-ups. IEEE Trans. Softw. Eng. **47**(3), 498–521 (2021)
14. Lewis, J., Fowler, M.: Microservices (2014). https://martinfowler.com/articles/microservices.html. Accessed 30 June 2021
15. Melegati, J., Chanin, R., Sales, A., Prikladnicki, R.: Towards specific software engineering practices for early-stage startups. In: Paasivaara, M., Kruchten, P. (eds.) XP 2020. LNBIP, vol. 396, pp. 18–22. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-58858-8_2
16. Melegati, J., Wang, X., Abrahamsson, P.: Hypotheses engineering: first essential steps of experiment-driven software development. In: IEEE/ACM Joint 4th International Workshop on Rapid Continuous Software Engineering and 1st International Workshop on Data-Driven Decisions, Experimentation and Evolution (RCoSE/DDrEE), pp. 16–19 (2019)
17. Ralph, P., Kelly, P.: The dimensions of software engineering success. In: Proceedings of the 36th International Conference on Software Engineering - ICSE 2014, no. 1, pp. 24–35. ACM Press, New York (2014)
18. Shenhar, A.J., et al.: Project success: a multidimensional strategic concept. Long Range Plan. **34**(6), 699–725 (2001)
19. Unterkalmsteiner, M., et al.: Software startups - a research agenda. e-Inform. Softw. Eng. J. **10**(1), 1–28 (2016)
20. Yli-Huumo, J., et al.: The Relationship Between Business Model Experimentation and Technical Debt, vol. 210, pp. 17–29 (2015)
21. Zaheer, H., Breyer, Y., Dumay, J., Enjeti, M.: Straight from the horse's mouth: founders' perspectives on achieving 'traction' in digital start-ups. Comput. Hum. Behav. **95**, 262–274 (2019)

2nd Workshop on Agility with Micro Service Programming

Towards Integrating Blockchains with Microservice Architecture Using Model-Driven Engineering

Simon Trebbau(✉), Philip Wizenty, and Sabine Sachweh

IDiAL Institute, University of Applied Sciences and Arts Dortmund,
Otto-Hahn-Straße 27, 44227 Dortmund, Germany
{simon.trebbau,philip.wizenty,sabine.sachweh}@fh-dortmund.de

Abstract. Blockchain presents a feasible method to persist immutable information in a distributed ledger to improve the level of authentication and trust. Moreover, smart contracts enable the automated execution of any contract concluded between participants of the Blockchain network. On the other hand, Microservice Architecture (MSA) is a novel approach towards service-based scalable applications. In our paper, we present an approach based on Model-Driven Engineering (MDE) that aims to facilitate the integration process of Blockchains into MSA-based applications in order to benefit from the advantages attributed to Blockchains.

Keywords: Microservice Architecture · Model-Driven Engineering · Code generation · Distributed ledger · Blockchain · Smart contract

1 Introduction

Blockchain constitutes a technology for information exchange and transactions that require a specific level of authentication and trust [17]. Additionally, a blockchain reduces the risk of data manipulation, system failure, and dependency at a single system component [9]. In recent years, blockchains emerged as an important and influential technology for businesses and society [9].

Moreover, modern blockchain technologies like Ethereum[1] also supports the usage of *smart contracts* [17]. Smart contracts act as autonomous agents in the blockchain network and contain program code that executes by a specific message from a user's transaction or another smart contract. Typical use cases for smart contracts are financial payments or contractual agreements.

Microservices Architecture (MSA), in which services are used as autonomously software building blocks, shares several similarities with the concept of smart contracts [16]. Both provide their functionalities over a specific interface. Also, they manage their own data and have an isolated deployment environment, e.g., a Kubernetes[2] pod or an Ethereum Virtual Machine (EVM).

[1] https://ethereum.org/.

[2] https://kubernetes.io/.

P. Gregory and P. Kruchten (Eds.): XP 2021 Workshops, LNBIP 426, pp. 167–175, 2021.
https://doi.org/10.1007/978-3-030-88583-0_16

MSA as well as smart contracts represent complex distributed systems. Furthermore, smart contracts can provide similar functionalities similarly to microservices, e.g., individual data storage and API [4]. Therefore, they can be used in a MSA application to increase authentication and trust [17].

This paper introduces an Model-Driven Engineering [2] (MDE)-based approach to ease the integration of blockchain technology into MSA-based applications. By using models as first class citizens in the development process, we abstract from implementation details to reduce the overall complexity. Additionally, the presented approach includes the usage of code generators to support the development process and increase code quality.

The remainder of this paper is structured as follows. Section 2 gives a brief introduction to blockchain technology and MDE of MSA. Section 3 introduces our MDE-based approach for the integration of blockchain technology into MSA. We validate the approach in Sect. 4. Section 5 provides related work. This paper concludes and outlines future work in Sect. 6.

2 Background

Blockchain. A blockchain is a shared digital and distributed ledger [10], which stores transaction data on multiple network nodes without a central party and according to an agreed policy. The involved nodes connect directly over a peer-to-peer network. Executed transactions are combined into blocks, which are linked together using cryptographic hash values [8]. For contributing a transaction to the blockchain, it is necessary to build consensus among all participants. The consensus is formed via a *consensus algorithm* [17] that synchronizes the distributed ledger on the different participant nodes.

Generally, a smart contract is an automated transaction protocol, which executes the terms of a contract [17]. Therefore, a smart contract represents a fragment of source code that could be executed automatically on a dedicated environment on the blockchain and perform various functionalities, e.g., a financial transaction or report at the end of an electric vehicle charging process.

Model-Driven Engineering of Microservice Architecture. A well researched approach to enable MDE for MSA is the Language Ecosystem for Modeling Microservice Architecture[3] (LEMMA) [11]. LEMMA provides a set of modeling languages and model transformations that are built using the Eclipse Modeling Framework (EMF) [15]. LEMMA utilizes methods and techniques from MDE to reduce the complexity of MSA engineering for various stakeholder groups, e.g., domain experts and microservice developers. The modeling languages provide the possibility to create models as an artifact in the software engineering process for enabling code generation and reasoning about microservice architectures.

LEMMA's Domain Data Modeling Languages (DDML) [13] enables the modeling of domain concepts and addresses the *domain viewpoint* on microservice

[3] https://github.com/SeelabFhdo/lemma.

architectures. DDML is used by domain experts and service developers to capture domain concepts in models and supports Domain-Driven Design (DDD) [5] patterns such as Entity, Aggregate, and Repository.

The Technology Modeling Language (TML) [13] provides a means for service operators and developers to construct models targeting technology-specific information for service implementation and operation. Therefore, TML allows capturing and modularization of information regarding programming languages, frameworks, and deployment technologies. Additionally, *technology aspects* [11] are supported as a concept for the integration of technology-specific metadata, e.g., database mappings or microservice interaction configuration, into the models.

LEMMA's Service Modeling Language (SML) [13] focuses on the *service viewpoint* on MSA. SML enables service developers to construct models for specifying the API of a microservice. In detail, the SML allows to specify interfaces including their data structure as well as interface dependencies on other microservices. To this end, SML models are able to import previously defined DDML models as well as other SML models.

The Operation Modeling Language (OML) [13] of LEMMA addresses the *operation viewpoint* on MSA and is used by service operators. OML encapsulates concepts for service deployment, e.g., deployment technologies, operation environments, service-specific configurations, and dependencies to infrastructure components.

In addition to the modeling languages, LEMMA also provides means to processing resulting models. Firstly, LEMMA contains *intermediate metamodels* and *intermediate model transformations* [7] to facilitate the processing of the constructed models. Based on these intermediate models, LEMMA includes a Model Processing Framework to ease the development of model processors like code generators, model analyzers, and model visualization.

3 A Model-Based Approach to Integrate Blockchain with Microservice Architecture

Our approach focuses on the research context of supporting the integration process of blockchain technology for MSA. For this purpose, we use MDE to abstract from implementation details to reduce the overall complexity by using microservice architecture viewpoint-specific modeling languages. Precisely, our approach provides the functionalities to realize the integration process of blockchain technology utilizing LEMMA's modeling languages.

With a view to MSA and blockchains as well as their potentially combination, some challenges arise. A specific challenge in MSA is the deployment and operation of the microservices [1], which also applies to blockchain because of their similarities. Furthermore, the handling of smart contracts also can be a challenging process because they need to be integrated into the application and blockchain [3]. Based on these challenges, the question is how can MDE be used in suitable places to abstract frequently occurring and possibly complex processes.

LEMMA-Based Integration of Blockchain Functionality into Microservice Architecture. Our approach uses LEMMA's modeling languages (cf. Sect. 2) to abstract from implementation details in MSA development to support the development process. Moreover, we use code generators to generate multiple artifacts, e.g., Java classes and configuration files. The presented approach focuses on the generation of blockchain-related artifacts like implementing the connection to the blockchain network. Figure 1 depicts our approach and the relation between the different LEMMA models, the code generators, and generated artifacts.

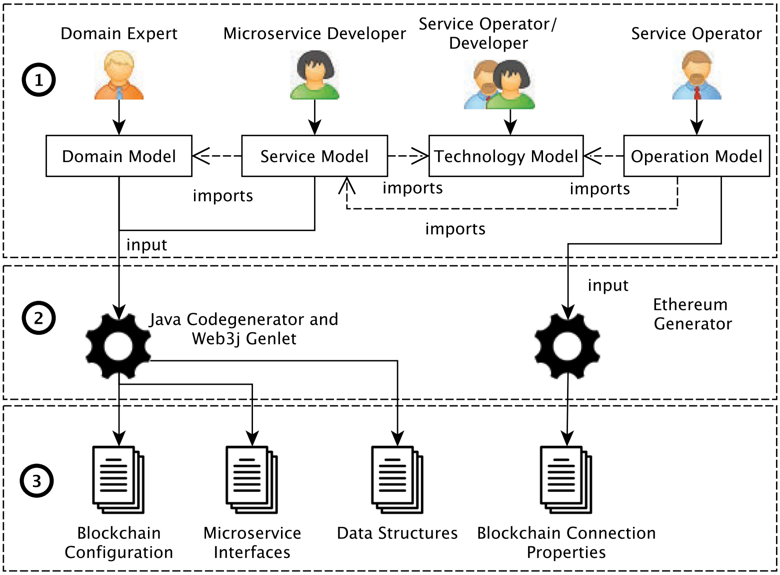


Fig. 1. LEMMA-based approach for integration of blockchain functionality into MSA.

The presented approach divides into three consecutive stages. Stage 1 comprises an agile modeling process by the stakeholder groups of MSA engineering. They collaborate to construct the models, which describe the MSA [11]. This approach considers all groups and their models, but with an increased focus on the models dealing with blockchain aspects of the application. It includes the `Domain Model` for data structures, the `Service Model` for configuration aspects or API definitions, the `Operation Model` for blockchain connectivity property initialization, and a `Technology Model` that defines Ethereum specific aspects like network properties.

Stage 2 utilizes the created models from Stage 1 as an input for the model processing. The `Java Base Generator` and `Web3j Genlet` use the `Domain` and `Service Model` to generate Java source code. *Genlets* are code generation modules introduced by LEMMA to generate individual source code for passed domain model and service model [12]. Additionally, the `Ethereum Generator` utilizes the `Operation Model` to derive blockchain service configuration properties for establishing the connection to a blockchain. Both the `Web3j Genlet` and the `Ethereum Generator` represent defined extensions of the LEMMA framework, which also could be extended by additional generation functions in the further if necessary.

Stage 3 shows the code generators created artifacts for the development process of the MSA application. The `Blockchain Configuration` configures communication with the blockchain network using the Web3j[4] library. It is generated as a separate artifact by the `Web3j Genlet`. It includes predefined java methods for connection establishment, transaction management, and adjustable methods for deployment and loading of smart contracts. Additionally, the `Microservice Interfaces` and `Data Structures` are generated based on the `Domain Model` and `Service model` to support the development process. The `Microservice Interfaces` can be used to trigger a smart contract by using the methods provided via the blockchain configuration artifact. Moreover, to enable the deployment process of the microservice in association with blockchain, the `Ethereum Generator` creates service-specific `Blockchain Connection Properties` for connecting the microservice to the blockchain.

4 Validation

This section validates the presented LEMMA-based approach for the integration of blockchain for MSA. For this purpose, we first introduce a case study as a basis for validation, followed by the results of our approach.

Case Study. To validate our approach, we introduce the PuLS[5] Park and Charge platform as a case study. The platform is being developed using LEMMA and a model-first approach and aims, among other things, to demonstrate the feasibility of our approach to ease the integration of blockchain functionality and MSA technologies in the context of MDE. The PuLS Park and Charge platform itself is being developed as part of an ongoing research project that aims to increase the availability of charging infrastructure for electric cars in inner-city areas. For this purpose, the platform allows citizens to share their private charging infrastructure with others. The architecture of the platform, which provides the sharing functionalities, is depicted in Fig. 2.

[4] https://github.com/web3j.

[5] https://parken-und-laden.de/.

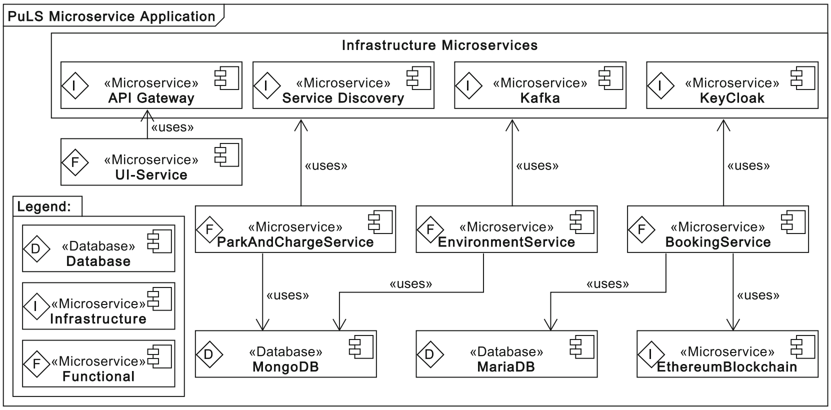

PuLS Microservice Application
Infrastructure Microservices
I «Microservice» API Gateway
I «Microservice» Service Discovery
I «Microservice» Kafka
I «Microservice» KeyCloak
«uses»
F «Microservice» UI-Service
«uses»
«uses»
«uses»
Legend:
D «Database» Database
I «Microservice» Infrastructure
F «Microservice» Functional
F «Microservice» ParkAndChargeService
F «Microservice» EnvironmentService
F «Microservice» BookingService
«uses»
«uses»
«uses»
«uses»
D «Database» MongoDB
D «Database» MariaDB
I «Microservice» EthereumBlockchain

Fig. 2. Microservice Architecture of the PuLS case study application.

The PuLS platform consists of three functional microservices. The ParkAndChargeService is responsible for sharing and managing the charging infrastructure for the citizen. To monitor the city's environmental data, the Environment Service collects data from IoT devices to keep track of particulate matter pollution. Bookings of the charging stations are realized via the BookingService. For ensuring the integrity of the bookings, the information is persisted in a blockchain. Storing the bookings increases citizens' trust, as the information can no longer be changed in the blockchain, ensuring that the process of using the charging stations is secured. The intention to incorporate blockchain at this point builds on the interest of PuLS project partners, who have proposed a corresponding integration in this context. Accordingly, a research context in the PuLS project is to find out how and for which possible use cases blockchain technology can be integrated into the Park and Charge platform. Our use case illustrates a corresponding opportunity.

To provide a better understanding regarding the possible modeling of blockchain-specific aspects using LEMMA, Listing 1.1 presents an excerpt of the booking operation model used by the Ethereum Generator to create the Blockchain Connection Properties for microservice deployment. Lines 1 to 6 describe the general deployment of the BookingService. The blockchain-specific configuration is defined via a technology aspect (cf. Sect. 2) in Lines 7 to 15. Depending on the use case, the technology model referenced here can be extended by any other specific aspects that also can be used in context of service and operation models. The shown EthereumNetwork aspect initializes a hostName and port for communicating with an Ethereum network node. Additionally, a gasLimit and gasPrice, which are necessary for transaction management are defined in the model. Furthermore, the privateKey associates an attribute for accessing an Ethereum wallet available on the addressed network node to execute or receive transactions.

```
...
@technology(container_base)
@technology(ethereum)
container BookingContainer
    deployment technology container_base::_deployment.Kubernetes
        deploys bookingService::v01.de.fhdo.BookingService
            depends on nodes ethereumOperation::Ethereum {
            aspects {
                ethereum::_aspects.EthereumNetwork(
                    privateKey="...",
                    hostName="http://localhost",
                    port=8545,
                    gasLimit=4712388,
                    gasPrice=20000000000
                ); }}
```

Listing 1.1. Excerpt of `BookingService` Operation Model.

Results. For our approach we provide a model representation of the PuLS architecture using LEMMA's modeling languages. Based on the resulting models, the `Ethereum Generator` and `Web3j Genlet` (cf. Fig. 1) create the code artifacts needed for the integration of basic blockchain functionality. In this context, the `Web3j Genlet` and the generation of the `Blockchain Configuration` demonstrate, that it is possible to support the integration of blockchain technology for MSA using MDE. Moreover, utilizing the `Ethereum Generator`, it was possible to abstract the deployment of microservices in association with blockchain. We use a basic lines-of-code metric to gain a first estimate of the efficiency of our approach by comparing the manually created models with the generated artifacts. This shows that the number of lines of generated code is higher than the number of lines of code needed to define the models. However, it should be noted that the metric refers to Java code. This may well produce different results for other programming languages. For replicability purposes, all artifacts for the approach are provided in a GitHub repository[6]. In addition to the LoC comparison, current research is working on other comparisons that provide a better idea of the generators efficiency. For the presented case study, the integration of the blockchain for MSA works successfully and is used in the presented PuLS research project in the development process.

5 Related Work

De Sousa *et al.* [14] presents an approach to constructing a prototype based on microservices and blockchain technology. It enables the integration and interaction between notary offices and other institutions, ensuring security and high speed in exchanging information between parties. As a case study, microservice architecture has been developed in which external institutions such as a hospital and post office and a notary's office interact with an Ethereum blockchain to manage a birth registration. In contrast to our approach, the integration of MSA and blockchain is related to a specific scenario and spares the usage of MDE.

[6] https://github.com/SeelabFhdo/xp2021.

Gorski and Bednarski [6] propose modeling support from the perspective of deploying distributed ledger solutions. Their approach uses MDE for transforming distributed ledger models into source code, e.g., deployment scripts or deployment configuration. Like in our approach, MDE is used to facilitate the development process of a distributed ledger. However, our approach addresses the integration of blockchain for MSA and focuses on blockchain as distributed ledger solutions.

6 Conclusion and Future Work

This paper has shown by means of a concrete example that it is feasible to integrate blockchain technology into MSA to increase authentication and trust (cf. Sect. 1) by using MDE. To this end, Sect. 2 introduced background information about blockchain and MDE of MSA. To support the integration of blockchain in MSA, we presented our MDE-based approach in Sect. 3. This approach utilizes LEMMA to generate Service-specific blockchain configurations. We validate the approach through a case study in Sect. 4. Additionally, we provide a brief overview of related work regarding MSA and blockchain (cf. Sect. 5).

For future research, we plan to diverge smart contracts program code from LEMMA's domain models. Additionally, we want to extend the existing code generators to provide better support for different blockchain technologies. Moreover, the code generator should also create blockchain deployment-related artifacts to enable the deployment process of blockchain components and its various network nodes. Another topic we will address relates to the compatibility of MDE with verification technologies. Also, a challenge which we are going to address in the future is the abstraction and modeling of relationships between microservices, the various blockchain network nodes and their user wallets.

References

1. Alshuqayran, N., Ali, N., Evans, R.: A systematic mapping study in microservice architecture. In: 2016 IEEE 9th International Conference on Service-Oriented Computing and Applications (SOCA), pp. 44–51. IEEE (2016)
2. Combemale, B.: Engineering Modeling Languages. Taylor & Francis, CRC Press, Boca Raton (2017)
3. Dannen C.: Solidity programming. In: Dannen, C. (ed.) Introducing Ethereum and Solidity, pp. 69–88. Springer, Berkeley (2017). https://doi.org/10.1007/978-1-4842-2535-6_4
4. Esposito, C., Castiglione, A., Choo, K.K.R.: Challenges in delivering software in the cloud as microservices. IEEE Cloud Comput. **3**(5), 10–14 (2016)
5. Evans, E.: Domain-Driven Design Reference, 1st edn. Dog Ear Publishing, Indianapolis (2015)
6. Gorski, T., Bednarski, J.: Applying model-driven engineering to distributed ledger deployment. IEEE Access **8**, 118245–118261 (2020). https://doi.org/10.1109/access.2020.3005519

7. Jézéquel, J.M., Combemale, B., Derrien, S., Guy, C., Rajopadhye, S.: Bridging the Chasm between MDE and the world of compilation. Softw. Syst. Model. **11**(4), 581–597 (2012)
8. Malik, S., Dedeoglu, V., Kanhere, S.S., Jurdak, R.: TrustChain: trust management in blockchain and IoT supported supply chains. In: 2019 IEEE International Conference on Blockchain (Blockchain). IEEE, July 2019. https://doi.org/10.1109/blockchain.2019.00032
9. Ølnes, S., Ubacht, J., Janssen, M.: Blockchain in government: benefits and implications of distributed ledger technology for information sharing (2017)
10. Quiniou, M.: Blockchain?: The Advent of Disintermediation. Wiley, Hoboken (2019)
11. Rademacher, F., Sachweh, S., Zündorf, A.: Aspect-oriented modeling of technology heterogeneity in microservice architecture. In: 2019 IEEE International Conference on Software Architecture (ICSA), pp. 21–30. IEEE (2019)
12. Rademacher, F., Sachweh, S., Zundorf, A.: Deriving microservice code from underspecified domain models using DevOps-enabled modeling languages and model transformations. In: 2020 46th Euromicro Conference on Software Engineering and Advanced Applications (SEAA). IEEE, August 2020. https://doi.org/10.1109/seaa51224.2020.00047
13. Rademacher, F., Sorgalla, J., Wizenty, P., Sachweh, S., Zündorf, A.: Graphical and textual model-driven microservice development. In: Bucchiarone, A., et al. (eds.) Microservices, pp. 147–179. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-31646-4_7
14. de Sousa, P.S., Nogueira, N.P., dos Santos, R.C., Maia, P.H.M., de Souza, J.T.: Building a prototype based on microservices and blockchain technologies for notary's office: an academic experience report, March 2020. https://doi.org/10.1109/ICSA-C50368.2020.00031
15. Steinberg, D., Budinsky, F., Paternostro, M., Merks, E.: EMF: Eclipse Modeling Framework, 2nd edn. Addison-Wesley, Boston (2008)
16. Tonelli, R., Lunesu, M.I., Pinna, A., Taibi, D., Marchesi, M.: Implementing a microservices system with blockchain smart contracts. In: 2019 IEEE International Workshop on Blockchain Oriented Software Engineering (IWBOSE), pp. 22–31. IEEE (2019)
17. Zheng, Z., Xie, S., Dai, H.N., Chen, X., Wang, H.: Blockchain challenges and opportunities: a survey. Int. J. Web Grid Serv. **14**(4), 352–375 (2018)

A Service Mesh for Collaboration Between Geo-Distributed Services: The Replication Case

Marie Delavergne(✉), Ronan-Alexandre Cherrueau, and Adrien Lebre

LS2N, Inria, Nantes, France
{marie.delavergne,ronan-alexandre.cherrueau,adrien.lebre}@inria.fr

Abstract. Edge computing is becoming more and more present, with sites geo-distributed around the globe. Applications on these infrastructures must be able to manage the latency and disconnections inherent to their distribution. One way to deal with these concerns could be to deploy one entire instance of the application per site and use a service mesh to manage the collaboration between the geo-distributed instances. More precisely, we propose to reify the location of application instances in REST requests and allow redirections between these requests thanks to a dedicated language and a service mesh allowing three types of collaborations. This paper focuses on the replication of a resource between multiple instances. Though it is still a work in progress, we demonstrated the relevance of our approach in the OpenStack ecosystem.

1 Introduction

Edge computing is getting more important, with more and more small datacenters at the edge of the network. Nonetheless, lots of applications do not benefit from the geo-distribution of wide-area networks and are not designed to handle the high latencies and disconnections implied by these distributions [8]. To deal with these concerns, we advocate for the placement of an instance of the application on each site. This way, each site is autonomous and can fully work if disconnected from the rest of the network [2]. Unfortunately, the collaboration is still missing: instances are able to function by themselves, but they cannot collaborate between each other and so do not benefit from the geo-distribution.

To provide such a collaboration without changing the code, we propose to leverage the service mesh concept. Service meshes help cloud computing applications solve different problems with their built-in functionalities. For example, to improve overall performance, load-balancing is provided. More largely, by intercepting communications, they provide functionalities to ease different operations, like traffic monitoring, access control, fault tolerance [3]. In general, they are implemented with proxies as sidecars for the services, without interfering with their code as they only work on requests passing from services to services. Their ability to intercept and redirect communications offers an opportunity to orchestrate requests between endpoints of any instance of the same application.

P. Gregory and P. Kruchten (Eds.): XP 2021 Workshops, LNBIP 426, pp. 176–185, 2021.
https://doi.org/10.1007/978-3-030-88583-0_17

In this paper, we propose Cheops, a service to use in combination with a service mesh to program on-demand collaborations between multiple instances of an application. To specify where a request will be executed at a fine-grained level, Cheops relies on the scope-lang proposal we initially developed [2].

Scope-lang extends applications API and allows the user to specify where (on which services) the request is executed. The language has been designed to provide different types of collaborations between application instances. For example, *sharing* is when a resource needed by a service has been created on another instance. This is the basic collaboration which allows to share resources between the instances. *Replication* allows operations on identical resources on different sites, to deal with availability of these resources in case of network partitioning or to improve overall performance. Finally, *cross* allows a resource to span across different sites. In this paper, we focus on *replication* (*sharing* has already been discussed [2], while *cross* is let as future work). By resources, we mean every entities managed by services, whether it is an entry in the database or something has complex and low-level as a network.

It is noteworthy that other frameworks or languages [4,7] have been proposed. However, they are invasive as they require to entangle geo-distribution in the business code. Non-invasive approaches generally follow the brokering approach: an entity is in charge of redirecting requests between the different instances. However, instances are not aware of the others. The goal of this proposal is to allow the instances to collaborate on-demand as if they were a single entity. Another way to see it is to allow the users to operate changes on resources on different sites through the API.

In this paper, we focus on how replication is interpreted and executed thanks to Cheops. We first explain scope-lang and how our general model works to allow DevOps to specify the location of a request execution. Then, we dive into the replication collaboration between different instances of the same service. In particular, how we manage replicas of a resource on different sites and how we handle disconnections, partitions or faults.

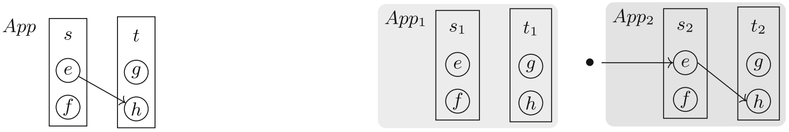
2 Scope-Lang, A Language to Reify the Geo-Distribution of Requests

In this section, we dive deeper into how scope-lang, Cheops[1] and our general model work together to allow collaborations outside of the application, and so keep a clear separation of concerns.

2.1 General Model

As a reminder, we have entire instances of the application on each site. Scope-lang parameters Cheops on a per-request basis in order to orchestrate collaborations between instances.

[1] https://gitlab.inria.fr/discovery/cheops.



(a) Application *App* made of two services s and t and four endpoints e, f, g, h. The $s.e \rightarrow t.h$ represents an example of a workflow.

(b) Two independent instances App_1 and App_2 of the *App* application. The • represents a client that executes the $s.e \rightarrow t.h$ workflow in App_2.

Fig. 1. Microservices architecture of a cloud application.

To explain collaboration between each instance of different services, let us take a look on how microservices based applications work. Each service composing the application exposes endpoints to communicate with other services. These endpoints are linked to a specific part of the business they achieve. When calling endpoints of other services, they form a workflow between services. For example, Fig. 1a shows an application *App* composed of two services s and t that expose endpoints e, f, g, h and one example of a workflow $s.e \rightarrow t.h$. Figure 1b shows the instantiation of the application *App* on two different sites and their corresponding service instances: s_1 and t_1 for App_1; s_2 and t_2 for App_2. A client (•) triggers the execution of the workflow $s.e \rightarrow t.h$ on App_2. It addresses a request to the endpoint e of s_2 which handles it and, in turn, contacts the endpoint h of t_2.

2.2 Scope-Lang

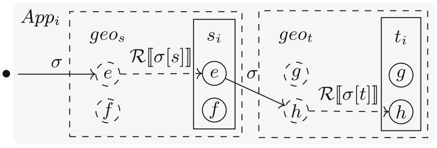
To parameter collaborations, we developed a domain specific language called scope-lang. A scope-lang expression (referred to as the *scope* or σ in Fig. 2a) contains location information that defines, for each service involved in a workflow, in which instance the execution takes place. The scope "$s : App_1, t : App_2$" intuitively means that the request must be achieved on the service s from App_1

$$\begin{array}{lll}
App_i, App_j & ::= \text{application instance} & \\
s, t & ::= \text{service} & \\
s_i, t_j & ::= \text{service instance} & \\
Loc & ::= App_i & \text{single location} \\
 & \;\;\mid\;\; Loc\&Loc & \text{multiple locations} \\
\sigma & ::= s : Loc, \sigma & \text{scope} \\
 & \;\;\mid\;\; s : Loc &
\end{array}$$

$$\mathcal{R}[\![s : App_i]\!] = s_i$$

$$\mathcal{R}[\![s : Loc\&Loc']\!] = \mathcal{R}[\![s : Loc]\!] \text{ and } \mathcal{R}[\![s : Loc']\!]$$

(a) scope-lang expressions σ and the function that resolves service instance from elements of the scope $\mathcal{R}$.



$\sigma = s : App_i, t : App_i$

(b) Scope σ interpreted by the geo-distribution service mesh *geo* during the execution of the $s.e \xrightarrow{\sigma} t.h$ workflow in App_i. Reverse proxies perform requests forwarding based on the scope and the $\mathcal{R}$ function.

Fig. 2. A service mesh to geo-distribute a cloud application

and t from App_2. The scope "t : *App₁*&*App₂*" specifies to execute the request on the service t of App_1 and App_2. Users set the scope of a request to specify the collaboration between instances they want for a specific execution. The scope is then *interpreted* by a dedicated module entitled Cheops during the execution of the workflow to fulfill that collaboration. The main operation it performs is *request forwarding*. To be more precise, reverse proxies in front of each service instance (geo_s and geo_t in Fig. 2b) intercept the request and interpret its scope to forward the request. "Where" exactly depends on locations in the scope.

The reverse proxy uses a specific function $\mathcal{R}$ (see Fig. 2a) to resolve the service instance at the assigned location. $\mathcal{R}$ uses an internal registry. Building the registry is a common pattern in service mesh using a *service discovery* [3].

In summary, scope-lang effectively parameters how Cheops will redirect the request. In the next section, we discuss how the replication is achieved.

3 Replication in Cheops

Replication is the ability to create and maintain identical resources on different sites: an operation on one replica should be propagated to the others, dealing with faults and disconnections and maintaining consistency based on our eventual model. Other consistency policies [1,10] could be envisioned, but let as future work as they do not change the general concept of scope-lang/Cheops. To get a better understanding of the point of replication, imagine a user who needs a huge resource (like an ISO image) both at home and at work. The resource can be replicated at creation on both sites and it will be the only time when the entire resource will go through the network. This saves a lot of bandwidth, and is especially useful if there is a partition between both sites.

3.1 Replication Model

Modular applications based on microservices usually follow a RESTful HTTP API. In most cases, they generate an identifier for each resource, which will be used by the API to retrieve, update or delete it. When receiving a request to create replicas, Cheops unify these identifiers with a data model called *replicant*.

A replicant is simply a meta-identifier we generate along with a mapping $site \rightarrow local\_identifier$. A replicant can thus be implemented for example as: $meta\_identifier : [site_n : local\_identifier_n, ...]$. We only store the location (site) of the replica and not the service used since it is possible to deduce the service with the incoming request. This is subject to change depending of the evaluation of our prototype. We could store also the involved service and/or the type of resource involved.

These replicants are stored in a database co-located to the Cheops agents. A copy of the replicant is stored on each site where its replicas are (the sites involved in the replication). Cheops has an API of its own to allow the user to check the state of operations, sites and inspect replicants.

3.2 Architecture Overview

Cheops agents are located on each instance site, with a reverse proxy besides every service transfering their requests to the agents. Agents communicate between each other and check each other status via heartbeats. Our implementation of Cheops uses Consul service mesh[2] and Envoy[3] as reverse proxy to intercept and redirect, when needed, the requests. It is also worth noting that Envoy intercepts inbound and outgoing requests from services except for requests coming from Cheops agents.

In Fig. 3, we represented the reverse proxy and Cheops as one single entity that intercepts the request as it is in Fig. 2b to ease the comprehension.

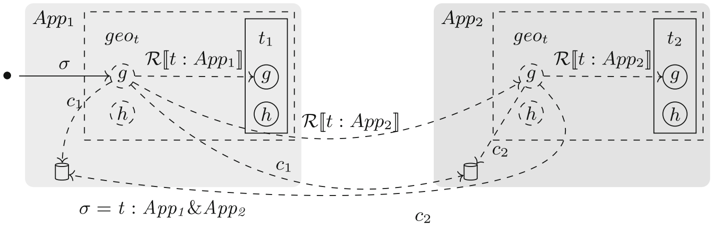

App_1
geo_t
t_1
σ
$\mathcal{R}[\![t : App_1]\!]$
g
h
c_1
$\mathcal{R}[\![t : App_2]\!]$
c_1
App_2
geo_t
t_2
$\mathcal{R}[\![t : App_2]\!]$
g
h
c_2
$\sigma = t : App_1 \& App_2$
c_2

Fig. 3. Modelling of the replication by forwarding on multiple instances. c_1 arrows represents Cheops agent on App_1 updates to the databases, c_2 arrows the one from Cheops agent on App_2.

3.3 CRUD Execution Workflow

First, to define what is the creation, update or delete workflow, we have to define what they do in our consistency model and what are their boundaries. The creation of resources replicated in an eventual consistency implies that every replicas are identical at creation and will be created eventually. The update of resources created with the replication in an eventual consistency implies that all replicas will be updated eventually, whether the user specifies a scope or not in its request. It is the same for deletes.

The operation obviously begins when the user makes the request. But for the end, we could consider that an operation ends either when there is one response and is returned to the user, or when the operation is executed on every sites. In an eventual consistency model, the latter *end* can come a lot later than the first response. It is important to know what happens in case of failure (partition, disconnection, server failure) during the execution until the first response, but also after, because the operation must be executed on our replicas at some point.

In eventual consistency, since the first response to arrive goes to the user before it might be applied everywhere, it is the responsibility of the user to

[2] https://www.consul.io/.
[3] https://www.envoyproxy.io/.

check with a request to Cheops or directly to involved services to know where the creation or updates are already applied. Users cannot assume because they received the answer the operation as already been applied everywhere.

Creation. The replication process to create a resource on App_1 and App_2 happens as follows:

1. A request for replication is addressed to the endpoint of a service of one application instance. For example in Fig. 3: $\bullet \xrightarrow{t:App_1 \& App_2} t.g$, where g is the endpoint for the creation of the resource managed by the service t.
2. The scope is extracted in the Cheops agent and the $\mathcal{R}$ function (from scope-lang) is used to resolve the endpoints that will store replicas. In Fig. 3: $\mathcal{R}[\![t : App_1 \& App_2]\!]$ is equivalent to $\mathcal{R}[\![t : App_1]\!]$ and $\mathcal{R}[\![t : App_2]\!]$. Consequently, t_1 and t_2 will be used for the resource creation.
3. The meta-identifier is generated and the replicant created using the meta-identifier and the location of execution. For example, if the generation yielded 72, we have: $\{$ $72 : [App_1 : none, App_2 : none]\}$. The replicant on App_1 (where the request was made) becomes the leader of the replicants. A log is created for future operations on replicas, as well to make sure the creation will be applied on every involved sites.
4. Each request is forwarded to the corresponding Cheops agent on involved sites and a copy of the replicant is stored in the database on those sites simultaneously. In Fig. 3: geo_t forwards the request to $t_1.g$ and $t_2.g$ and stores the replicant $\{72 : [App_1 : none, App_2 : none]\}$ in App_1 and App_2 databases. In the figure, this is represented by the c_1 arrows going to the cylinders.
5. Each contacted service instance executes the request and returns the results to their local Cheops agent, which updates the replicant with the local identifier. In Fig. 3: t_1 and t_2 return their local identifier, e.g., 42 and 6.
6. Cheops agents then proceed to propagate the updated information to other agents involved. In parallel, they send the entire response to the Cheops agent that stores the leader replicant. In Fig. 3: the replicant is now $\{72 : [App_1 : 6, App_2 : 42]\}$ on App_1 and App_2 sites databases, thanks to the updates represented by c_1 and c_2 arrows.
7. When the agent where the leader is receives the first creation response, it transfers it to the user who asked for the replication, replacing the local ID with the replicant meta ID.

Read. The process of reads is straightforward; to access a specific resource, users must either be on a site where one of its replicas is or specify in the scope on which location a replica of the resource to read is.

Update. From now on, every request made to update (or delete) is filtered to check if the id given corresponds either to a replicant meta identifier or a local replica identifier. The process is quite similar to the creation, but does not generate a new replicant or change an existing one. It only applies an update to replicas.

1. A request for an update of a previously created replica is addressed to the endpoint of a service of one application instance.
2. Cheops checks if the ID in the request exists in a replicant. If not, the request is sent back to the service to be executed. If it is, the requests is transferred to the Cheops agent storing the replicant leader. It gets the corresponding replicant to find every replicas (and thus sites) involved. The operation is stored in its log.
3. The request is copied as many times as necessary (with the corresponding local identifier) and sent to the Cheops agent of involved sites.
4. Local Cheops agents send the request to the corresponding service on their site, which executes the request normally.
5. Each Cheops agent sends back the response to the Cheops agent where the replicant leader is.
6. This agent sends back the response to the user, once again, with the meta-identifier where the local-identifier would be expected to notify the user that the replicas were updated.

Delete. As for the update, a delete on replicas can be identified either by a local identifier or the meta identifier. The process is identical as the update's.

3.4 Dealing with Faults

We define a fault as: a partition of an involved site, or a failure from this site, whether it is shut down, out of order, or if the request cannot be executed for any reason (not enough memory to create a resource for example).

It is also important to mention that if the site where the user sent its request is faulty (does not work in any way), the request obviously cannot be executed. The user can make the request to a more distant site.

Moreover, the "during an operation" can refer to two distinct phases. As we discussed before, the end of an operation can be seen as: when a replica has been created/updated/deleted and the user has been notified, and when the operation is applied to all replicas. So "during an operation" is between the request of the user and before one of these end. In our consistency model, this conveys no difference to the process.

If a site fails where a replica is supposed to be, other Cheops will be informed due to its heartbeat (or rather lack of). Any other operation received by the leader will then be retried according to the log when the site comes back again. Therefore, a site is considered to be eventually available again unless it is removed. If a site is removed from the system, every replicant that were hosting a replica on this site must delete the site from their mapping (from the replicant). The leader will be in charge of this particular task.

Faults during operations The operation will be applied *eventually* on all involved sites. This eventual consistency uses a consensus protocol, and in our case, an implementation of Raft [6]. For example, the leader's log allows to replay operations that are not yet applied. It is the responsibility of the Cheops agent where the leader is to ensure that operations are applied eventually.

Faults while there are replicas When a site fails while there are replicas somewhere without any particular operation running, no heartbeat is received by other Cheops agent and the replica is considered unavailable temporarily.

If a site where a replica is was partitioned at some point but could be used locally, only read queries can be made, and these reads might be stale. When rejoining the cluster, operations will be applied on the site so it is up-to-date thanks to the leader's log.

4 Discussion

In this paper, the service mesh and proxy mentioned were respectively Consul and Envoy, but this approach could work with other proxies or services meshes available such as Istio[4] or Open Service Mesh[5]. The approach differs from using a service mesh to redirect the requests with load-balancing or in case of failures by giving the users the ability to chose where their requests will be executed per-demand.

The users are thus responsible to trigger the request based on their needs and the availability of sites. In the case of a infrastructure such as OpenStack, this means that it gives back the DevOps the ability to decide where a request will be available. But for more common applications, the user might not need as much information about the execution of their request. In this case, it is then totally possible to apply usual quality of service techniques available in a service that would execute the request by adding a relevant scope itself.

4.1 Proof of Concept

Though Cheops is still a work in progress, we demonstrated the relevance of sharing a resource in a proof of concept (PoC) on OpenStack [2]. This PoC gives DevOps the ability to make multiple independant instances of OpenStack collaborative. Using our approach with OpenStack would allow to manage a geo-distributed infrastructure as a usual IaaS platform. This is a breakthrough as several initiatives tried to propose a framework to manage edge infrastructures and processes [5,9], but due to the difficulty of delivering a software as complex/complete as OpenStack, the work to be redone would be colossal.

4.2 Limitations

There are of course some limitations to our approach. First, it requires microservices-based applications that exposes an API for services communications. These applications need to be able to work on a single site since we will

[4] https://istio.io/.

[5] https://openservicemesh.io/.

deploy them autonomously on every sites. Moreover, every instance of the application should have the same version for identical resources.

This approach ensures consistency at the service-level, but for the resources they manage. The only operations available to manipulate these resources are therefore the ones exposed by the API. Thus, the resources are maintained as identical as the API allows it, but nothing less. For example, nothing can be said about the consistency of two VMs booted through this process; their internal state will probably diverge, as expected.

5 Conclusion

In this paper, we presented the replication mechanisms of Cheops and how scope-lang allows its parametrization. The ultimate goal of this project is to allow generic collaborations between multiple instances of the same application without applying intrusive changes in the business code. We presented especially the different workflows for the replication collaboration.

As future work, we identified other collaboration mechanisms that could be relevant. For example, our replication strategy could be extended in order to include a controller and propose an abstraction similar to the ReplicaSet and its controller in the Kubernetes ecosystem[6]. The point would be to add control loop capabilities into Cheops in order to maintain the desired number of replicas according to the infrastructure changes. We could also propose different ways to keep the consistency between replicas, giving more choice for the users (e.g., giving them the choice to change the location of a replica if its site fails).

Besides replication, additional collaborations can also be envisioned (such as an *otherwise* operator that will ask for a request to be executed on a specific site and if this one is unavailable, execute on the other specified). Any future implementation will depend on the needs observed when deploying this solution.

Acknowledgments. We would like to thank Matthieu Juzdzewski and Arnaud Szymanek for their work on Cheops.

References

1. Akkoorath, D.D., et al.: Cure: strong semantics meets high availability and low latency. In: 2016 IEEE 36th International Conference on Distributed Computing Systems (ICDCS). IEEE (2016)
2. Cherrueau, R.A., Delavergne, M., Lebre, A., Rojas Balderrama, J., Simonin, M.: Edge Computing Resource Management System: Two Years Later! Research Report RR-9336, Inria Rennes Bretagne Atlantique (2020)
3. Li, W., et al.: Service mesh: challenges, state of the art, and future research opportunities. In: 2019 IEEE International Conference on Service-Oriented System Engineering (SOSE), pp. 122–1225 (2019)

[6] https://kubernetes.io/docs/concepts/workloads/controllers/replicaset/.

4. Martin, B., Prosperi, L., Shapiro, M.: An environment for composable distributed computing. In: EuroDW 2020–14th EuroSys Doctoral Workshop (2020)
5. Mortazavi, S.H., Salehe, M., Gomes, C.S., Phillips, C., de Lara, E.: CloudPath: a multi-tier cloud computing framework. In: Proceedings of the Second ACM/IEEE Symposium on Edge Computing, pp. 1–13 (2017)
6. Ongaro, D., Ousterhout, J.: In search of an understandable consensus algorithm. In: 2014 {USENIX} Annual Technical Conference, {USENIX}{ATC} 2014 (2014)
7. Safina, L., Mazzara, M., Montesi, F., Rivera, V.: Data-driven workflows for microservices: genericity in Jolie. In: 2016 IEEE 30th International Conference on Advanced Information Networking and Applications (AINA). IEEE (2016)
8. Satyanarayanan, M.: The emergence of edge computing. Computer **50**(1), 30–39 (2017)
9. Wang, N., et al.: ENORM: a framework for edge node resource management. IEEE Trans. Services Comput. **13**, 1086–1099 (2017)
10. Zhu, Y., Wang, Y.: SHAFT: supporting transactions with serializability and fault-tolerance in highly-available datastores. In: 2015 IEEE 21st International Conference on Parallel and Distributed Systems (ICPADS), pp. 717–724 (2015)

Implementation of a Microservice-Based Certification Platform

Sebastian Copei[1](✉), Manuel Wickert[2], and Albert Zündorf[1]

[1] Kassel University, Kassel, Germany
{sco,zuendorf}@uni-kassel.de
[2] Fraunhofer IEE, Kassel, Germany
manuel.wickert@iee.fraunhofer.de

Abstract. Interoperability between edge devices and backend systems plays an important role for a successful digital transformation of the energy system. To provide interoperability, edge devices as well as backend systems use standardized communication protocols. Vendors of such systems often certify their products to verify compatibility to a certain version of communication protocol standard. Since standardization and certification processes are classically very time consuming, they seem to be incompatible with agile software development. Therefore we proposed an agile standardization and certification approach. Here we will present a basic implementation of a microservice based certification platform that supports certification as a service. The platform allows software vendors to integrate certification of their products directly in their continuous delivery pipeline. This will lay the ground for the evaluation of certification as a service in a concrete scenario.

Keywords: Microservices · Standardization · Certification · Agile

1 Motivation

The transformation of the energy system is becoming a key driver of the digitalization of the energy sector. On the one hand, digitalization supports a real-time view of all producers and consumers of the complete energy system. Therefore smart meters will play an important role. On the other hand, digitalization will enable the compensation of fluctuations, caused by wind and photovoltaic (PV) feed-in, by aggregating huge numbers of controllable consumers and energy storages, e.g. electrical vehicles (EVs).

One big challenge in digitalization is the connectivity to such edge devices. In contrast to the end-consumer sector, the energy sector uses edge devices with significantly longer planned lifetimes. Furthermore, backend systems and edge devices are often built from different vendors. Therefore interoperability between these devices and backend systems is very important. To provide interoperability, standardized communication protocols (e.g. IEC 61850 [3], IEC 60870-5-104

S. Copei and M. Wickert—These two authors contributed equally.

P. Gregory and P. Kruchten (Eds.): XP 2021 Workshops, LNBIP 426, pp. 186–191, 2021.
https://doi.org/10.1007/978-3-030-88583-0_18

[2], OPC UA [5]) are common for such communication scenarios. Edge devices will typically be certified to verify that they are compliant with a given version of the standard. However, standardization and certification processes are classically very time-consuming tasks and therefore not compatible with agile development processes. In our last work [1] we proposed a new standardization and certification approach that can be integrated into agile development workflow for edge devices as well as for backend systems.

In Horizon 2020 project Interconnect the communication between edge devices as well as backend systems shall be enabled by SAREF. SAREF is an ontology to enable interoperability between smart devices in the energy sector. The SAREF Standard will be extended and used during the project to provide a connection between EV-charging stations, home energy management systems (including smart meters), and superordinate operation systems. To enable agile development of the standard extensions as well as all the edge devices and backend components, we implemented a certification as a service platform. This platform should be used within the project to verify that each component is compliant with a certain version of the SAREF standard.

In Sect. 2 we give a short introduction to our agile certification approach from [1]. The Sect. 3 describes our certification as a service platform. The evaluation and the next steps are described in Sects. 4 and 5.

2 Agile Certification

In contrast to the state-of-the-art standardization and certification process, we proposed an agile standardization and certification approach. The basic idea is to make small adaptions to existing versions of a communication standard. The small adaptions lead to newly published versions of the standard. This will increase the number of standards but also decrease the time to integrate new requirements into a communication standard. With a new communication standard version, a few conformance tests should be adapted or created. The conformance tests will be used to verify that components are compliant with the adapted version of the standard. In Fig. 1 the agile process is shown by two interwinded cycles. On the left side, the standardization process is shown. The right side shows the implementation of a conformant component like an edge device. The certification of the component builds the bridge between both cycles.

From the perspective of the software development process of edge devices or backend systems, the certification should be part of a continuous integration or deployment process (CI/CD). Therefore the certification has to be fully automated, including conformance tests, evaluation, and the issuance of the certificate. A certification body may offer all these functionalities by a certification service. The software vendor by themselves may use this service by calling it from a CI/CD Pipeline (see Fig. 2).

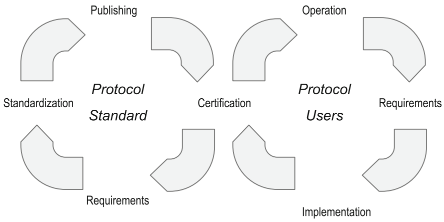


Fig. 1. Agile Standardization Process

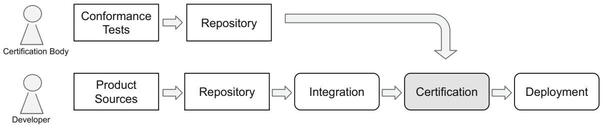


Fig. 2. Continuous certification pipeline

3 Certification Platform

To support certification bodies and to provide certification services, we developed a certification platform. The platform is implemented based on our proposed microservice architecture in [1]. We implemented five backend services in *NestJS* and a frontend service in *Angular*. The platform is developed as open-source and is available on GitHub[1]. For the communication between the backend services, we used *Apache Kafka*. The user management is realized with *Keycloak*. As mentioned in [1] the certification platform takes a software artifact during the test phase of a build process and performs compliance tests of a standard.

For our implementation, we used *Docker* images to provide an artifact. So a smart energy device, e.g. an electrical charging station control software, could be delivered as a container to our platform. Therefore we provide a *Docker* image to exchange software artifacts between the CI Pipeline and the Certification platform. During the test phase of the CI Pipeline, the certification platform is called via a REST API. Besides the *Docker* image the version of the communication standard is provided to declare which compliance tests should be performed. The artifact service starts a *Docker* container of the provided image. Afterward, the test suite service runs the containers for certification. All running and started containers are managed by the job runner service. If all tests passed, the certification services signs the provided *Docker* image from the CI Pipeline with a signature to ensure the correctness of the certificate. Finally, the signed container is provided back to the pipeline. This enables an agile certification during agile software development.

In Fig. 3 we summarize the current platform and services. As mentioned before, there are services for managing artifacts, tests, certificates, and runnable test environments. The latter service also provides the API for the CI integration.

[1] https://github.com/sekassel-research/caas-platform.

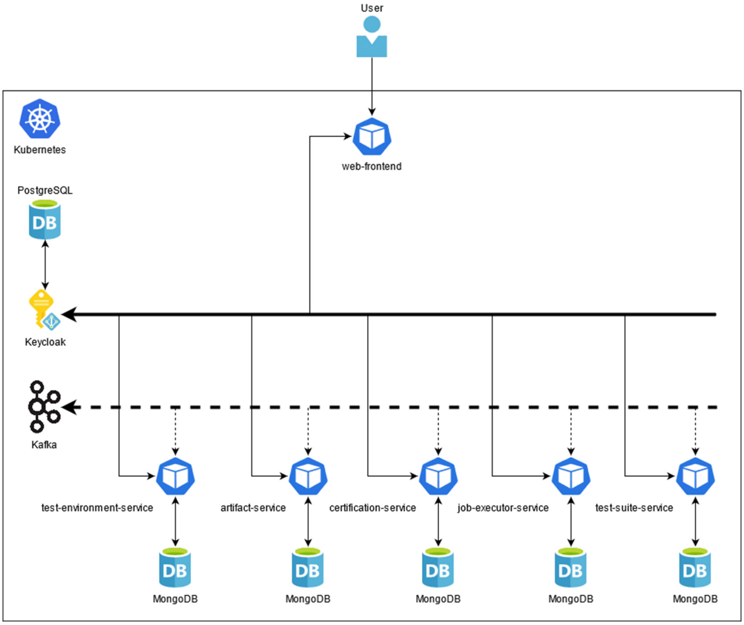


Fig. 3. CaaS platform

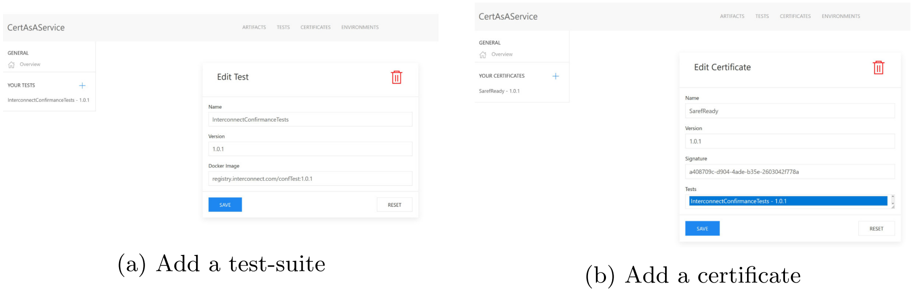
4 Evaluation

The current implementation allows providing artifacts, test suites, certificates, and runnable test environments. These features are tested with integration tests. In addition, it is possible to start a single artifact with a single test-suite to achieve a certificate. For the evaluation of this workflow, we created a simple scenario from the context of the interconnect project.

The GoodEnergy AG wants to implement a new service for the interconnect project. The be interoperable with the other system components, our SAREF communication extension has to be implemented. After the developer of the GoodEnergy AG provides a version of their service, they want to use the certification service to test and certify their service for usage in the interconnect project.

The certification service is provided by a project internal certification body. The certification body will use our platform and a test suite with all conformance tests to provide this service. Figure 4(a) shows the newly created test-suite. To use this test-suite in the further process, the certification body needs to map the standard version number to the first created test-suite. This is shown in Fig. 4b.

After the certification body creates a test suite and maps it to a version number, the GoodEnenrgy AG can use the certification service. Figure 5a shows how software can be uploaded via UI, instead of using the REST API directly from the CI/CD Pipeline. In the next step, you choose the version number, for which certification has to be performed. Figure 5b shows a created test environment for



YOUR TESTS

InterconnectConfirmanceTests - 1.0.1

Edit Test

Name: InterconnectConfirmanceTests

Version: 1.0.1

Docker Image: registry.interconnect.com/confTest:1.0.1

(a) Add a test-suite

YOUR CERTIFICATES

SarefReady - 1.0.1

Edit Certificate

Name: SarefReady

Version: 1.0.1

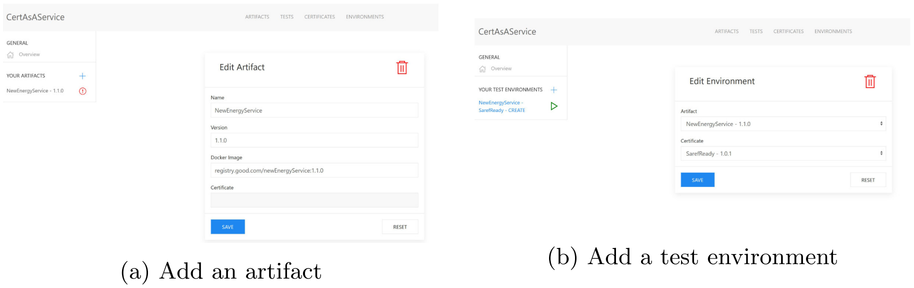
Signature: a408709c-d904-4ade-b35e-2603042f778a

Tests: InterconnectConfirmanceTests - 1.0.1

(b) Add a certificate

Fig. 4. Actions as a standardization committee

the uploaded software and version number. Finally, the conformance tests can be performed, tests will be evaluated and the certificate can be issued. Here a certification corresponds to a signed Docker image. The certification service signs the image, so it is possible to verify that the current image content is compliant with a certain standard version.



YOUR ARTIFACTS

NewEnergyService - 1.1.0

Edit Artifact

Name: NewEnergyService

Version: 1.1.0

Docker Image: registry.good.com/newEnergyService:1.1.0

Certificate:

(a) Add an artifact

YOUR TEST ENVIRONMENTS

NewEnergyService - SarefReady - CREATE

Edit Environment

Artifact: NewEnergyService - 1.1.0

Certificate: SarefReady - 1.0.1

(b) Add a test environment

Fig. 5. Actions as GoodEnergy AG

The UI certificate download is currently a work in progress. Therefore the development of a process can not completely be evaluated for the charging station. However, the scenario shows how the developed platform will support agile certification of communication standards by charging stations as an example.

5 Future Work

The current implementation does not contain a relevant number of test cases for the charging station certification. Therefore we plan to implement these tests and evaluate the certification platform during the further development of smart components. Furthermore, we want to implement the possibility to provide a system or a composition of multiple services as an environment to test a service interaction within a system. Finally, we want to evaluate the tool within an industry case study from the energy domain.

6 Acknowledgement and Disclaimer

This Publication is part of a project [4] that has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement N°857237. The sole responsibility of this publication lies with the author. The European Union is not responsible for any use that may be made of the information contained therein.

References

1. Copei, S., Wickert, M., Zündorf, A.: Certification as a service. In: Paasivaara, M., Kruchten, P. (eds.) Agile Processes in Software Engineering and Extreme Programming - Workshops, pp. 203–210. Springer, Cham (2020)
2. IEC 60870-5-104 - Telecontrol equipment and systems. Standard, International Electrotechnical Commission, Geneva, CH (2006)
3. IEC 61850 Standard Series, Communication networks and systems in substations. Standard, International Electrotechnical Commission, Geneva, CH (2020)
4. Interconnect project - homepage. https://interconnectproject.eu/. Accessed 20 Apr 2021
5. OPC Unified Architecture, IEC 62541, Standard Series. Standard, OPC Foundation, International Electrotechnical Commission, Scottsdale, USA (2008)

Poster Presentations

Multiple Roles of Middle Managers in Agile Project Governance: An Activity Theory Perspective

Maduka C. Uwadi(✉)

School of Psychology and Computer Science, University of Central Lancashire,
Preston, UK
mcuwadi@uclan.ac.uk

Abstract. Project governance (PG) is an important activity in agile software development (ASD) projects. Middle managers (MMs) are part of the governance structure in ASD projects. PG and middle management phenomena in ASD projects are under-researched and not fully understood. This ongoing study aims to fill a gap by investigating the roles of MMs in agile PG through the lens of Activity Theory. The study adopts a qualitative and interpretive case study approach. To date, the study has identified 24 roles that MMs perform during agile PG.

Keywords: Agile project governance · Middle managers · Agile software development · Activity Theory · Case study

1 Purpose

Project governance (PG) is an important but complex activity performed during agile software development (ASD) projects, and encompasses the necessary oversight, processes, tools, manpower, and support for project accomplishment. It is the "framework, functions, and processes that guide project management activities in order to create a unique product, service, or result to meet organizational strategic and operational goals" [14, p. 4]. In relation to agile projects, PG is under-researched and not fully understood [7,11].

Middle managers (MMs) occupy the middle-level position in an organisation's governing structure and they link senior management (SM) with the lower-level workforce [2]. In ASD projects, MMs function within agile teams [6]. The MMs are expected to work alongside agile teams and play their role to ensure delivery of ASD projects, hence they are part of the project governance structure therein. However, there is evidence that the role of MMs in ASD projects is not clearly defined and not fully understood [3,6,12]. Agile projects are considered lightweight, self-organising, and flexible, therefore practitioners question how 'management' and 'governance' fit in. Middle management role uncertainty is one of the top ranked challenges affecting agile teams [3]. Such uncertainty provokes tensions within agile teams during task execution [6], thereby threatening team stability and project congruity.

P. Gregory and P. Kruchten (Eds.): XP 2021 Workshops, LNBIP 426, pp. 195–200, 2021.
https://doi.org/10.1007/978-3-030-88583-0_19

This ongoing study (part of a wider PhD study) seeks to answer the question: **What are the roles of middle managers in agile project governance?** The study is investigating the PG activity in an ASD project in order to determine the roles of MMs in agile PG within a selected organisation. Presentation of the organisation's agile PG practices is not within the scope of this article.

2 Research Design

This research employs a qualitative and interpretive case study design, which is well-suited because it puts the researcher in the world of the participants living the PG and middle management experience, thereby allowing him to interpret their views and experiences [16].

Due to the complex and multifaceted nature of agile PG, given that it involves multiple actors, processes, mechanisms, tools, and socio-technical interactions [11], the research demanded a flexible socio-technical theoretical framework with expansive analytical and interpretive power to aid analysis of agile PG and mid-management. Activity Theory (AT) lends itself to these demands [4,8,9].

According to AT [9], an activity is comprised of six main components, viz., *subject*, *tools*, *object*, *rules and norms*, *community of significant others*, and *division of labour*. By considering these components, AT enables analysis of activities in breadth and depth [4]. In the present study, AT was used as the principal theory to develop an Activity-oriented Project Governance (APGov) theoretical framework to support the research (Fig. 1). The APGov framework[1], which also incorporates other theories and studies [5,10,11,13,15], was developed to aid case study data collection, analysis, and interpretation of findings. The unit of analysis for this study is the PG activity, which has ASD project as the main governance object, and middle management as one of the activity actors.

A single case study (a research limitation) involving a Nigerian fintech holding company (HOLDCOY) was undertaken. Data was collected between February and March 2020, comprising nine semi-structured interviews, three team meeting observations, company documentation, and questionnaires for collection of company and project profile information. The company and its participants were selected using convenience and stratified sampling. The company was selected because (a) it is a technology-enabled organisation that practices PG, (b) it has used agile methods to execute and govern software development projects for eight years, and (c) its organisational structure includes middle management. Participants included SM, MMs, and lower-level workforce, with agile methods experiences ranging from 8 months to 11–12 years[2]. The researcher limited the study to analysis of the PG activity and middle management in one of HOLDCOY's divisions; TECHCOY division, which is the agile team executing the ASD project under examination.

Six interviews out of the nine transcribed interviews in HOLDCOY, as well as organisational structure document, email and instant messaging correspondence,

[1] Description of framework components is available at https://bit.ly/3ijFCcx.

[2] For more sample population and data sources details, visit https://bit.ly/3ijFCcx.

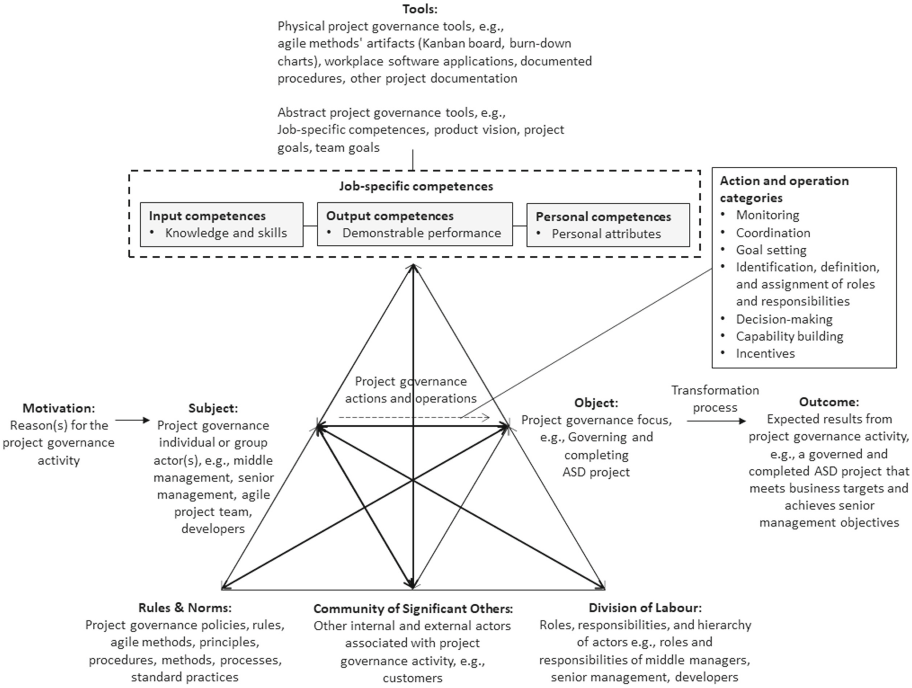
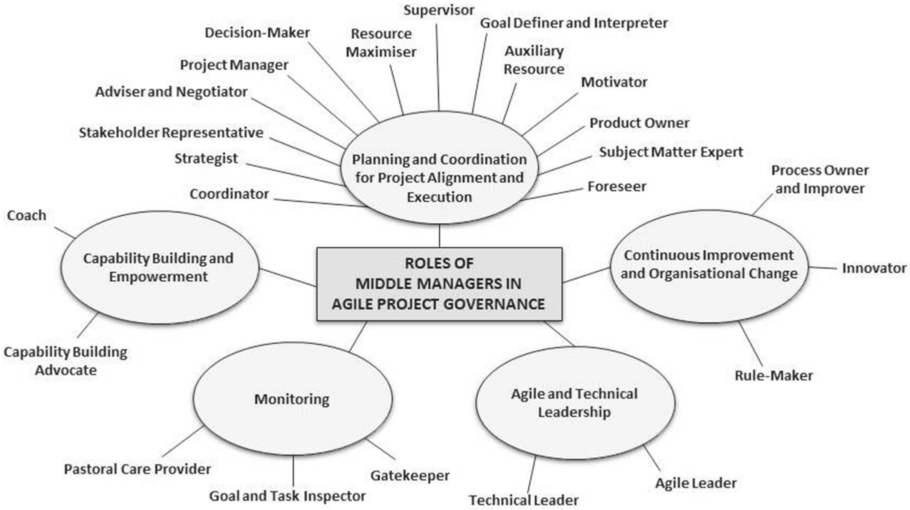

Tools:
Physical project governance tools, e.g., agile methods' artifacts (Kanban board, burn-down charts), workplace software applications, documented procedures, other project documentation
Abstract project governance tools, e.g., Job-specific competences, product vision, project goals, team goals
Job-specific competences
Input competences
• Knowledge and skills
Output competences
• Demonstrable performance
Personal competences
• Personal attributes
Action and operation categories
• Monitoring
• Coordination
• Goal setting
• Identification, definition, and assignment of roles and responsibilities
• Decision-making
• Capability building
• Incentives
Project governance actions and operations
Motivation: Reason(s) for the project governance activity
Subject: Project governance individual or group actor(s), e.g., middle management, senior management, agile project team, developers
Object: Project governance focus, e.g., Governing and completing ASD project
Transformation process
Outcome: Expected results from project governance activity, e.g., a governed and completed ASD project that meets business targets and achieves senior management objectives
Rules & Norms: Project governance policies, rules, agile methods, principles, procedures, methods, processes, standard practices
Community of Significant Others: Other internal and external actors associated with project governance activity, e.g., customers
Division of Labour: Roles, responsibilities, and hierarchy of actors e.g., roles and responsibilities of middle managers, senior management, developers

Fig. 1. APGov framework

and profile questionnaire responses, have been analysed thus far using thematic network analysis (TNA) for thematic analysis [1]. The six interviews produced 192 pages of transcripts, which were read several times and coded by applying a coding framework in line with the TNA process [1]. NVivo software was used to organise text segments into codes, which later formed themes for the construction of a thematic network interpreting various roles of MMs in agile PG.

3 Findings

Preliminary findings are presented as a thematic network (Fig. 2) comprised of (a) basic themes, which are the lowest-order premises contained in the data, (b) organising themes, which are higher-order themes (categories of grouped basic themes) summarising main discoveries contained in the data [1], and (c) global theme, which is the superordinate theme that encapsulates "the principal metaphors in the data as a whole" [1, p. 389]. In the thematic network, 24 basic themes, which represent MM roles in the agile PG activity's division of labour, are grouped into five organising themes (role categories), which are linked to the global theme - *Roles of middle managers in agile project governance.* Participants affirm that MMs play pivotal roles in agile PG practice, thereby helping to accomplish their project. Findings will be of benefit to agile teams with MMs.


ROLES OF MIDDLE MANAGERS IN AGILE PROJECT GOVERNANCE
Planning and Coordination for Project Alignment and Execution
Supervisor
Goal Definer and Interpreter
Decision-Maker
Resource Maximiser
Auxiliary Resource
Project Manager
Motivator
Adviser and Negotiator
Product Owner
Stakeholder Representative
Subject Matter Expert
Strategist
Foreseer
Coordinator
Coach
Capability Building and Empowerment
Capability Building Advocate
Continuous Improvement and Organisational Change
Process Owner and Improver
Innovator
Rule-Maker
Monitoring
Pastoral Care Provider
Goal and Task Inspector
Gatekeeper
Agile and Technical Leadership
Technical Leader
Agile Leader

Examples of Interview Quotes:

Motivator role: "…So those are things I try putting in place as incentives for, so even apart from promotion, these are just things that within our division we're trying to do to ensure that yes the team is always happy, and the team is always motivated week in, week out to achieve results" (MM).

Coordinator role: "…their role is actually very critical…the project middle managers are responsible to work closely with all the internal teams and the contacts at the external team to ensure that every single deliverable as stated in the business requirement documentation…are completed, tested and delivered" (SM).

Strategist role: "…I find every strategic way to ensure that we achieve this go live at the shortest time possible through incremental delivery…which the agile methodology gives us the permission to do" (MM).

Fig. 2. Thematic network of MM roles in agile PG and interview quote examples

4 Research and Practice Implications

This research exemplifies and advances the use of AT in agile research, and adds to studies on agile PG and MMs in ASD projects, which are limited. Organisations that use agile methods and have MMs can use the model of MMs' roles as a tool for (a) creating job descriptions and person specifications for recruitment of MMs, (b) education and training, and (c) ensuring MMs maintain acceptable levels of job performance when governing and delivering ASD projects.

5 Contributions

This study introduces an APGov theoretical framework by applying AT to agile PG and middle management. The study is also developing a model of MMs' roles in agile PG. The model will describe multiple roles that MMs perform when working alongside agile teams and governing ASD projects. The model will help SM, MMs, aspiring MMs, agile teams, and researchers to better understand the roles of MMs in agile PG practice, which may lead to stronger organisation-project strategic connections and project success, better working relationships between MMs and teammates in agile teams, and further research.

Acknowledgements. I would like to thank my PhD supervisors, Dr. Peggy Gregory, Prof. Ian Allison, and Prof. Helen Sharp, for their continued support. We thank the Agile Research Network (ARN) for funding this study.

References

1. Attride-Stirling, J.: Thematic networks: an analytic tool for qualitative research. Qual. Res. **1**(3), 385–405 (2001)
2. Balogun, J.: From blaming the middle to harnessing its potential: creating change intermediaries. Br. J. Manag. **14**(1), 69–83 (2003)
3. Barroca, L., Dingsøyr, T., Mikalsen, M.: Agile transformation: a summary and research agenda from the first international workshop. In: Hoda, R. (ed.) Agile Processes in Software Engineering and Extreme Programming – Workshops. XP 2019. LNBIP, vol. 364, pp. 3–9. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-30126-2_1
4. Crawford, K., Hasan, H.: Demonstrations of the Activity Theory framework for research in Information Systems. Australas. J. Inf. Syst. **13**(2), 49–68 (2006)
5. Crawford, L.: Senior management perceptions of project management competence. Int. J. Proj. Manag. **23**(1), 7–16 (2005)
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. Gregory, P., Barroca, L., Sharp, H., Deshpande, A., Taylor, K.: The challenges that challenge: engaging with agile practitioners' concerns. Inf. Softw. Technol. **77**, 92–104 (2016)
8. Iyamu, T.: A case for applying Activity Theory in IS research. Inf. Resour. Manag. J. (IRMJ) **33**(1), 1–15 (2020)
9. Karanasios, S.: Framing ICT4D research using Activity Theory: a match between the ICT4D field and theory? Inf. Technol. Int. Dev. **10**(2), 1–17 (2014)
10. Kujala, J., Aaltonen, K., Gotcheva, N., Pekuri, A.: Key dimensions of project network governance and implications to safety in nuclear industry projects. In: EURAM 2016: Manageable Cooperation? (2016)
11. Lappi, T., Karvonen, T., Lwakatare, L.E., Aaltonen, K., Kuvaja, P.: Toward an improved understanding of agile project governance: a systematic literature review. Proj. Manag. J. **49**(6), 39–63 (2018)
12. 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.) Agile Processes in Software Engineering and Extreme Programming – Workshops. XP 2019. LNBIP, vol. 364, pp. 13–19. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-30126-2_2
13. Nyandongo, K.M., Khanyile, K.: Governance arrangements for agile projects. In: Proceedings of the International Conference on Industrial Engineering and Operations Management, pp. 23–26 (2019)
14. PMI: Governance of Portfolios, Programs, and Projects: A Practice Guide. Project Management Institute (2016)
15. Vlietland, J., van Vliet, H.: Towards a governance framework for chains of Scrum teams. Inf. Softw. Technol. **57**, 52–65 (2015)
16. Walsham, G.: Interpretive case studies in IS research: nature and method. Eur. J. Inf. Syst. **4**(2), 74–81 (1995)

Cherry Picking - Agile Software Development Teams Applying Design Thinking Tools

Franziska Dobrigkeit(✉), Christoph Matthies, Philipp Pajak, and Ralf Teusner

Hasso Plattner Institute, University of Potsdam, Potsdam, Germany
franziska.dobrigkeit@hpi.de

Abstract. Design Thinking (DT) is an established approach to conceptualize software products before starting the product development work. Research suggests that software development can benefit from a continuous integration of DT throughout Agile development processes. However, practitioners and researchers lack an in-depth understanding of which tools from the ever-growing DT toolbox are suited to support software development teams and their processes and how these tools can be applied to the teams' daily work. As initial steps towards closing this knowledge gap, we present our experiences from testing five different DT tools from a previously developed toolbox with four Agile software development teams. Each team chose three tools to apply to their product, problem, and context during a workshop. We present summarised findings regarding the use cases, benefits, and challenges of these tools as experienced by the participants. Overall, the teams welcomed the DT tools and were able to independently apply them to achieve the desired effects, e.g., to highlight user needs, find product issues, and discover team challenges.

Keywords: Design Thinking · Agile · Software development teams

1 Purpose

Research on Design Thinking (DT) indicates that it can positively impact software development activities, e.g., by facilitating a deeper understanding of the product, the users, and their needs [1,2,15,16], and increasing team collaboration [2,4]. Accordingly, various researchers propose integrating aspects of DT with modern Agile software development [9,11,17]. While the research is often focused on DT as an approach to support initial software design phases, some researchers suggest continuous use through all stages of software development [7,17]. Within such approaches, DT tools benefit Agile teams in later process stages, aiding during their ongoing development activities [5]. However, there is still a lack of understanding of which DT tools can support agile development teams. Accordingly, our research focuses on empowering Agile team members and DT novices with suitable DT tools and evaluating their experiences and collected perceptions (cmp. [5,6]). Towards this goal, we developed

P. Gregory and P. Kruchten (Eds.): XP 2021 Workshops, LNBIP 426, pp. 201–206, 2021.
https://doi.org/10.1007/978-3-030-88583-0_20

the *DT@IT Toolbox* in previous research [8,13]: a collection of 12 DT tools targeted at supporting software development teams in their work (freely available for download at https://hpi-epic.github.io/dt-at-it-toolbox/).

2 Research Method

Within this study, we employ a convergent mixed method research design [3] to collect empirical evidence for a total of 5 of the tools from the DT@IT Toolbox. We evaluate the tools with four teams from German Small and Medium Enterprises (SMEs). We selected the teams based on company size, self-reported use of an Agile development process, low self-reported knowledge of DT, and co-location to allow for an in-person workshop at the company site. We met with each teams' lead to introduce our toolbox and research and asked them to select three of the twelve tools for their team based on what made the most intuitive sense for their current use cases and problems. Next, we met at the team's site and conducted a three-and-a-half-hour workshop, in which the teams applied each of the three chosen DT tools to issues or topics relevant to their project. We introduce the tools with the help of the worksheets from the DT@IT Toolbox. Each worksheet includes information on required prerequisites, the time needed, an explanation of the tool, and examples from a software development context. The workshops were structured as follows:

1. 15 min introduction to Design Thinking and our research
2. 60 min Application of first DT Tool
   (a) Handing out the worksheet from our toolbox
   (b) Answering comprehension questions
   (c) Application of the DT tool to a problem or topic of the teams choice
   (d) Filling out a tool evaluation survey
3. 60 min Application of second DT tool (detailed steps see 2. a–d)
4. 60 min Application of third DT tool (detailed steps see 2. a–d)
5. 15 min Group interview.

During the workshops, we took part as participant observers and employed four forms of data collection to capture the participants' experience with the DT tools: The facilitating researcher took field notes on the teams' process including, use-case chosen, discussions in the team, results, challenges. Additionally, the researcher took pictures of the resulting artifacts, e.g., sketches or whiteboards. At three points of the workshop, we asked participants to fill out a survey to evaluate the DT tool they had just experienced. Items in the survey asked for the general experience with the tool, what they liked and disliked, further possible and impossible use-cases, and feedback on the worksheet. At the end of each workshop, we conducted a semi-structured group interview asking them to provide feedback on the workshop, the worksheets, and the tools. Thus, we carried out four semi-structured group interviews lasting between 15 to 20 min. The facilitating researcher took notes during each interview. After conducting all workshops, we analyzed our notes from observations and interviews and the survey answers. We iteratively coded and clustered the observation notes, the interview notes, and the survey responses to derive connections, patterns, and juxtapositions [10]. Thus we derived the perceived benefits, perceived challenges, and use cases presented in Sect. 3.

3 Findings

We evaluated five of the twelve toolbox tools, namely A Beginners Mind, the Customer Journey Map, 30 s Sketch, 5 Whys, and Five Finger Feedback. Table 1 depicts the use cases, benefits, and challenges we identified for each tool from our empirical data. Overall the participating teams welcomed the additional tools and the different working styles. The toolbox worksheets enabled the participating teams to independently apply the tools and achieve the desired outcomes, e.g., discovering product issues, generating and discussing ideas, or learning something about their product, users, and team members. These findings are in line with similar research on UX methods [14] and suggested benefits of DT [1,2,4,12,15,16].

Table 1. Summarized findings for each DT tool evaluated as part of this study

Use cases	Benefits	Challenges
A beginner's mind		
• feature design • UI design • framework choice • architecture choice • algorithm understanding	• explaining things simply • better understanding of problem or topic • abstraction of problem & technical knowledge	• too abstract for some • a beginner is required
Customer journey map		
• find & tackle UX issues • document features • document acceptance tests	• easy to take the user's perspective • helps uncover flaws in UI and process flow	• personas are required • finding the right scope • required time • keeping artifacts updated
30 second sketch		
• GUI-focused problems • searching for different perspectives	• easy to do • easy to dispose • fast results	• low confidence in drawing skills might be a barrier
Five whys		
• finding core problem during bug fixing	• allows to dig deeper than intuitive first answer • sensitizes for simplified answers & root causes	• might feel invasive to the questioned person
Five finger feedback		
• personal feedback • with managers • on a task or project • group feedback • in retrospectives • after longer meetings	• easy to do • easy to remember • feedback on different aspects	• accepting feedback • distinguishing feedback categories can be hard • difficult with broad topic • not suitable for large groups

4 Research and Practice Implications

The results from this study support our hypothesis that DT tools can benefit Agile teams during later stages of their development process, as they are aiding in their ongoing development activities and challenges. However, we were only able to evaluate five selected DT tools with a limited number of teams. Accordingly, further research with additional tools and teams is necessary to provide more evidence. Additionally, we only evaluate the direct effects of DT tool usage, and more research is required to collect and assess long-term results.

For Agile practitioners, we provide a low-friction possibility to get acquainted with suitable tools with our DT toolbox and the accompanying materials. The use cases and benefits identified within this research can guide the selection of fitting tools for a given situation, and the identified challenges provide hints of what to prepare beforehand or what to be aware of during the application of a specific tool. Our toolbox can serve as a stand-alone support for software development teams. In this case, teams that want to further their DT knowledge beyond the application of single tools will require additional in-depth explanations on DT and the associated concepts and background. Alternatively, our toolbox can support more holistic integration approaches (cmp. [9,11,17]) by providing DT use cases and tools to apply in later development stages.

5 Contributions

We contribute empirical evidence on how and when a selection of five DT tools can support Agile software development teams in their daily work. We summarize the employed use cases and the benefits and challenges experienced by the groups for each tool. The participating teams independently and successfully applied the introduced DT tools with the help of worksheets from the DT@IT Toolbox [8]. Thus, our findings suggest that DT novices can benefit from applying single DT Tools without extensive introduction or training in DT concepts. Furthermore, our results indicate that one-time use of the DT tools already provided the reported benefits. However, these are most likely not permanent without repeating similar activities. Additionally, the use of single DT tools only provides a limited perspective on DT as it does not support developing a DT mindset or an understanding of the DT process. Nonetheless, we conclude that the evaluated worksheets from the DT@IT toolbox and the identified use cases provide a low-barrier entry-point to DT, allowing agile development teams to gain first-hand experiences with DT tools and experiencing their benefits.

References

1. Brown, T.: Design thinking. Harvard Bus. Rev. **86**(6), 84 (2008)
2. Clark, K., Smith, R.: Unleashing the power of design thinking. Des. Manag. Rev. **19**(3), 8–15 (2008)
3. Creswell, J.W., Clark, V.L.P.: Designing and Conducting Mixed Methods Research. Sage Publications, Thousand Oaks (2017)
4. De Paula, D., Dobrigkeit, F., Cormican, K.: Design thinking capability model (DTCM): a framework to map out design thinking capacity in business organisations. In: 15th International Design Conference, pp. 557–566 (2018)
5. Dobrigkeit, F., de Paula, D.: Design thinking in practice: understanding manifestations of design thinking in software engineering. In: Proceedings of the 27th ACM Joint Meeting-ESEC/FSE 2019, pp. 1059–1069 (2019)
6. Dobrigkeit, F., de Paula, D., Carroll, N.: InnoDev workshop: a one day introduction to combining design thinking, lean startup and agile software development. In: 2020 IEEE 32nd Conference on Software Engineering Education and Training (CSEE&T), pp. 1–10. IEEE (2020)
7. Dobrigkeit, F., de Paula, D., Uflacker, M.: InnoDev: a software development methodology integrating design thinking, scrum and lean startup. In: Meinel, C., Leifer, L. (eds.) Design Thinking Research. Understanding Innovation, pp. 199–227. Springer, Cham (2019). https://doi.org/10.1007/978-3-319-97082-0_11
8. Dobrigkeit, F., Pajak, P., de Paula, D., Uflacker, M.: DT@IT toolbox: design thinking tools to support everyday software development. In: Meinel, C., Leifer, L. (eds.) Design Thinking Research. Understanding Innovation, pp. 201–227. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-28960-7_13
9. Grossman-Kahn, B., Rosensweig, R.: Skip the silver bullet: driving innovation through small bets and diverse practices. Lead. Des. **18**, 815–829 (2012)
10. Harding, J.: Qualitative Data Analysis from Start to Finish. Sage, Thousand Oaks (2013)
11. Hildenbrand, T., Meyer, J.: Intertwining lean and design thinking: software product development from empathy to shipment. In: Maedche, A., Botzenhardt, A., Neer, L. (eds.) Software for People. Management for Professionals, pp. 217–237. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-31371-4_13
12. Holloway, M.: How tangible is your strategy? How design thinking can turn your strategy into reality. J. Bus. Strategy **30**(2/3), 50–56 (2009)
13. Pajak, P., Dobrigkeit, F.: DT@IT toolbox: initial release. Version v1.0 (2021). https://doi.org/10.5281/zenodo.4602920
14. Pedersen, T.Ø.: UX toolbox for software developers. Ph.D. thesis (2016)
15. Porcini, M.: Your new design process is not enough-hire design thinkers! Des. Manag. Rev. **20**(3), 6–18 (2009)
16. Ward, A., Runcie, E., Morris, L.: Embedding innovation: design thinking for small enterprises. J. Bus. Strategy **30**(2), 78–84 (2009)
17. Ximenes, B.H., Alves, I.N., Araújo, C.C.: Software project management combining agile, lean startup and design thinking. In: Marcus, A. (ed.) Design, User Experience, and Usability: Design Discourse. LNCS, vol. 9186, pp. 356–367. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-20886-2_34

From Project to Product

Matthew Philip[1](✉) and Yoan Thirion[2]

[1] St. Louis, USA
[2] Luxembourg, Luxembourg

Abstract. As technical advances have enabled organizations to deliver software to the market faster, in turn shortening the feedback loop for new ideas and spurring innovation, legacy organizations need to update their mindset from a project-driven to a product-driven approach or risk being displaced by product-native organizations. This poster shows the high-level principles that represent our experience guiding organizations with a project-to-product approach.

Keywords: Product · Project · Outcomes · Experimentation · Value · Flow · Vision

1 History and Background

Organizations that are product-native – that is, product-oriented from their beginning – typically do not need to take a project-to-product journey, inasmuch as they have never known a time when IT was considered anything but integral to the organization's success. On the other hand, organizations who have inherited a legacy approach to information technology such that they have traditionally regarded IT as a cost center and separate from the business often experience the existential threat of being disrupted (or worse) and therefore choose to orient or reorient themselves toward products and product development, resulting in a more or less intentional rethinking of their ways of working.

For example, one of the organizations in our experience had a highly matrixed, globally distributed IT group of more than 12,000 people that optimized for project-management concerns rather than its customers and the flow of value to them. This organization had a history of so-called "Agile" and "DevOps" transformations, but none of them clearly dealt with moving from project to product. These transformations yielded some gains but ultimately, because they were internally oriented, did not focus on true customer and user outcomes.

2 Defining "Product"

A digital product (and therefore product management) is fundamentally different from a project (and project management). According to Jez Humble, some assumptions of *projects* can include:

- Once we've built it, it doesn't change much

P. Gregory and P. Kruchten (Eds.): XP 2021 Workshops, LNBIP 426, pp. 207–212, 2021.
https://doi.org/10.1007/978-3-030-88583-0_21

- In the course of building it, we don't learn much significant information
- You must complete it because you can start using it.

In contrast, digital products [3]:

- Will change a lot over their lifecycle
- Allow us, in the course of building them, to discover large amounts of information that can inform decisions and directions
- Can be used and provide value before they are "complete".

A working definition of product is (adapted from the Scrum Guide):

> A product is a vehicle to deliver value. It has a clear purpose, functional boundary, stakeholders, users and customers. A product could be a single or a group of services, physical products, applications, platforms, systems, and data.

3 Principles for Moving Toward Product-Orientation

As technical advances have enabled organizations to deliver software to the market faster, in turn shortening the feedback loop for new ideas and spurring innovation, legacy organizations need to update their mindset from a project-driven to a product-driven approach or risk being displaced by product-native organizations.

This change in mindset is significant, covering a wide range of concerns from psychological safety to budgeting to success measures. This poster shows the high-level principles that represent our experience guiding organizations with a project-to-product approach.

In moving toward product-orientation, we prefer the following principles:

3.1 Outcomes over Outputs

Outputs can be defined as delivering features, user stories or other work item types without respect to whether they are the right things to build or make a difference in user success or perceived value. A project mindset prioritizes outputs with metrics such as conformance to plan and feature delivery. A feature delivered that does not produce a desired outcome may still be considered a success in a project-oriented approach. However, by preferring *outcomes* – that is, measurable impact toward user and/or customer goals – we will acknowledge that not every feature we plan will necessarily achieve what we hoped it would but instead optimize for user success. It is the difference between simply doing work and doing the right work.

As Barry O'Reilly writes [4]:

> An outcome-based approach… demands that you clarify the success you seek—in quantifiable terms—by crisply defining what you're trying to achieve so people know why it matters… Clarity of destination allows people to explore different options to discover if the investments you make are moving you in the direction you desire… An outcome-based approach allows you to be accurate, more adaptive, and take action to course-correct and own the results you gather relative to the final destination.

3.2 Solving Problems over Building Solutions

Solving problems orients the team toward the user/customer. In particular, a product-orientation enlists engineers and cross-functional teams in generating the ideas for features rather than merely carrying out the solutions of a single product owner. John Dewey wrote that "a problem well put is half solved" [5]. Product-minded teams should build competency in and spend time identifying the problems they're trying to solve, as this will improve their chances of building the right things.

3.3 Options over Requirements (and Optionality over Linearity)

By framing potential work as options rather than requirements (which carry the connotation that they must be done), product teams emphasize their ability to discard unhelpful work in the face of new information. Teams make plans when they have less information than they will discover, so even the language of "requirement" can inhibit the ability to change course.

As Donald Reinertsen writes [6]:

> Fast feedback loops give us the ability to truncate unproductive paths quickly, which unlocks resources for other purposes … In contrast, the traditional approach to development focuses on up-front planning rather than adaptation. We tried to forecast everything and failed to do this accurately.

3.4 Experiments over Backlogs (and Hypotheses over Features)

Ultimately, product or service development is a process to test an hypothesis about system behaviour in the environment or market it is developed for [12]. Project-minded teams start planning with feature ideas, whereas product-minded teams plan with hypotheses. The language of experimentation expresses the uncertainty inherent in software development and changes the definition of success from features that are implemented but not validated (and therefore uncertain progress) to validated learning and true progress.

3.5 Customer-Validated Learning over PO Assumptions

This is why it is important to take a user-centered approach. We found that the vast majority of project-oriented teams did not use experience-design and design-thinking techniques such as personas; product-oriented teams did.

According to Daniel Vacanti, "True business value can be determined only after delivery to the customer. Choices about what to work on and when, then, are really just you placing bets on what you think the customer will find valuable" [11]. This statement emphasizes the limitation on a single product owner's knowledge and ability to forecast the future.

3.6 Measuring Value over Measuring Cost

Project-oriented organizations often prioritize measurements such as scope, cost and time. However, we have found, like Humble, that these concerns are fundamentally unsuited for product management inasmuch as the success of a product isn't dependent on these factors. Or, as Humble says, "How much it costs doesn't matter if people don't send you money" [3]. Additionally, Douglas Hubbard found that the concern is unwarranted and can actually mislead: "Even in projects with very uncertain development costs, we haven't found that those costs have a significant information value for the investment decision… The single most important unknown is whether the project will be canceled. …The next most important variable is utilization of the system, including how quickly the system rolls out and whether some people will use it at all" [1]. An obsession with cost undermines a focus on outcomes and experimentation.

3.7 Flow over Utilization

Reinertsen found that "Capacity utilization increases queues exponentially" and that "operating at high levels of capacity utilization increases variability". As a result, he recommends that product teams control queue size and not capacity utilization [7].

3.8 Product Vision, Strategy, Personas and Principles over Product Roadmaps

Product vision, strategy, personas and principles (aka product manifesto) enable the experimentation and problem solving referred to earlier, whereas roadmaps connote a fixed scope-and-time-based approach. In a product-oriented team, we use the former – which typically don't change as much – to guide development.

3.9 Small-Batch Delivery over Big-Batch Delivery

Reducing batch size has many quantifiable benefits that support product thinking, including reduced cycle time, faster feedback, increased employee motivation and reduced variability in flow [7].

3.10 Optimizing for Assumptions Being Wrong over Optimizing for Assumptions Being Right

Product-oriented teams embrace the reality that most of their ideas will not work. One of the reasons we de-emphasize traditional feature-based roadmaps is because "at least half of our ideas are just not going to work" [2]. Kohavi found that "evaluating well-designed and executed experiments that were designed to improve a key metric, only about one-third were successful at improving the key metric!" [9]. Additionally, "Netflix considers 90% of what they try to be wrong" [8]. This reality requires a strategy to deliver and validate as quickly as possible.

3.11 Teams of Missionaries over Teams of Mercenaries

Product-team culture is perhaps best described by John Doerr's "missionaries and mercenaries" metaphor. "Mercenaries are driven by paranoia; missionaries are driven by passion… Mercenaries focus on their competitors and financial statements; missionaries focus on their customers and value statements … missionaries are obsessed with making a contribution … and… are fundamentally driven by the desire to make meaning" [10].

3.12 Business-Driven over IT- or PMO-Driven

The movement from project to product requires not only business involvement and collaboration, but business orientation, subordinating IT and the Project-Management Office, which are often driven by project-management concerns that undercut product success discussed earlier.

References

1. Douglas Hubbard. https://www.cio.com/article/2438748/the-it-measurement-inversion.html
2. Cagan, M.: Inspired: How to Create Tech Products Customers Love. Wiley, Hoboken (2017)
3. Jez Humble. https://lectures.leanagile.pm/
4. Barry O'Reilly. https://barryoreilly.com/explore/blog/your-mission-is-to-produce-outcomes-not-outputs/
5. Dewey, J.: Logic, the Structure of Inquiry. Henry Holt & Co, New York (1938)
6. Reinertsen, D.: Towards Developing Accelerators in Half the Time. https://accelconf.web.cern.ch/ipac2011/papers/weib02.pdf
7. Reinertsen. http://lpd2.com/sample-page/the-principles-of-flow/
8. Moran, M.: Do it wrong quickly: how the web changes the old marketing rules (2007)
9. Kohavi, R., et al. http://ai.stanford.edu/~ronnyk/2013%20controlledExperimentsAtScale.pdf
10. Doerr, J.: Two Kinds of Internet Entrepreneurs, UPenn. https://knowledge.wharton.upenn.edu/article/mercenaries-vs-missionaries-john-doerr-sees-two-kinds-of-internet-entrepreneurs/
11. Vacanti, D.: Actionable Agile Metrics For Predictability: An Introduction Kindle Edition (2105)
12. Barry O'Reilly. https://barryoreilly.com/explore/blog/how-to-implement-hypothesis-driven-development/

Panels

The Stories We Tell: Experience, Research, or Patterns?

Dennis Mancl[1] (✉) and Steven D. Fraser[2]

[1] MSWX Software Experts, Bridgewater, NJ 08807, USA
dmancl@acm.org
[2] Innoxec, Santa Clara, CA, USA

Abstract. Several "story" formats are used by software researchers and practitioners to document research results and share best practices. Research papers are the staple of software conferences and journals: papers report the results of research projects and a wide range of empirical experiments. Experience reports and software patterns are two alternative formats to share results and propagate knowledge and best practices. Experience reports relate experience and cautionary tales while software patterns distill experience into a compact form. An XP2021 panel session orchestrated by Steven Fraser – featuring panelists Ademar Aguiar, Casper Lassenius, Mary Lynn Manns, Ken Power, and Rebecca Wirfs-Brock – discussed how "story" formats each have a role to advance the practice of software engineering.

Keywords: Innovation · Experience reports · Research papers · Software patterns

1 Stories Relating Research and Practitioner Experience

Throughout the 75-year history of software research and development, researchers and practitioners have advanced the field by sharing key learnings and best practices. Sharing is achieved through written documents: books and papers that document academic research, industrial experience, successes, failures, and retrospectives on the application of new methods and tools. Recently, sharing has also included internet-accessible audio and video recordings such as podcasts and YouTube videos. In both cases (oral and written), the software community's goal is to contribute knowledge that others may leverage and extend.

The panel discussed three formats for sharing best practices: experience reports, research papers, and patterns. Each of the three formats is characterized by its own stylistic guidelines. Experience reports are personal stories written in the first person to document a personal journey. Research papers are accounts of researchers applying the scientific process to a specific technical investigation. Patterns are descriptions of problem and solution pairs within a general framework. Practitioners and researchers select an appropriate style to share their ideas and results with the software community.

P. Gregory and P. Kruchten (Eds.): XP 2021 Workshops, LNBIP 426, pp. 215–221, 2021.
https://doi.org/10.1007/978-3-030-88583-0_22

2 Research Papers: Validating Research

Each academic field and many technical conferences have "guidelines" for research papers [1]. Casper Lassenius (Aalto University) related an informal set of guidelines for effective research papers. Casper (XP2021 academic track co-chair) shared his "COVID-D" research reporting model:

- Context and Contribution
- Objectivity
- Validation
- Interesting
- Depth
- Delightful

Each element of the model is an essential piece of a successful research paper's content and structure. The **Context** and **Contribution** of a research paper explain the essential elements of the research work; **Objectivity** requires that both successes and failures are reported; **Validation** references data to support the paper's claims and conclusions; a paper should be **Interesting** to both researchers and practitioners; **Depth** means that the paper contains enough detail for understanding and insight; **Delightful** means that the authors use an engaging style to attract and sustain readership.

In Casper's opinion, the Context of a research paper should include both academic context and practitioner relevance. Authors hope to interest readers in the acceptance (and possible adoption) of their paper's results, so the "academic context" is a matter of illustrating how results complement work reported by other researchers. Industry practitioners are motivated to read papers based on contextual research applicability.

Casper also noted that originality is an essential ingredient of a research paper. Authors must explain, identify, and distinguish (as unique) their contributions to the field. To validate research claims, the authors should include supporting data, from experiments – based on either new or community results. It was also observed that research papers generally focus on a very narrow set of questions. The requirements of validation and depth make it almost impossible for a paper to answer complex questions within journal or conference page limit constraints.

When we write research papers in the field of software engineering, we hope that our papers are interesting enough that others will build on our work. It is a continuing challenge to find an audience for research. The panel moderator, Steve Fraser (Innoxec), reflected on this issue. He recalled a complaint from Fred Brooks at an OOPSLA 2007 panel session on software engineering [2]. Brooks was worried that we don't understand enough about others' successes and failures. His point: "I know of no other field [software] where people do less study of other people's work."

3 Experience Reports: Personal Stories

Rebecca Wirfs-Brock (Wirfs-Brock Associates) and Ken Power (Independent Software Engineer) explained that experience reports are not as objective as a research paper, because an author relates their personal story – rather than reporting validated experimental results. Experience reports may also share emotional perspectives and feelings.

Ken believed that practitioners should write more experience reports. Ken noted, however, that all of us are challenged “finding the time to write down our stories and get them clear in our heads.” While the community benefits from experience reports, authors are often inexperienced writers, and they depend on conference assigned “shepherds” to relate and document their story. Shepherds are the authors’ guides though the storytelling challenges at conferences such as ACM SPLASH, XP, and Agile [3][4]).

A shepherd or writing coach is an essential catalyst as Ken explained, “As an author, [the shepherd is] somebody in your corner who wants you to succeed, who wants you to tell your story in the best possible way – so you’re not staring at that blank page on your own.”

4 Patterns: Distilling Many Stories

Mary Lynn Manns (Fearless Change) and Ademar Aguiar (University of Porto) explained the writing process for patterns. Patterns are short one- to two-page documents focused on practical solutions to a problem. A well-written pattern takes time to develop, and the writing task can be quite difficult. Mary Lynn observed, “A good patterns author will take years to get one good pattern written on a particular topic.” A draft pattern needs to be verified, and a solid process to check a pattern is to interview other experts and collect more stories. As a draft pattern evolves, it becomes more general and addresses a wider range of contexts.

Patterns are best if they are short and simple. Ademar emphasized two more important properties in telling good stories – Accessibility and Reusability – and in good patterns, both beginners and experts will have access to knowledge that is reusable.

A pattern is not based on novelty. Ademar quoted Brian Foote: “Patterns are a blatant disregard of originality.” A pattern author does not create new knowledge, rather their goal is to record existing knowledge in a useful and reusable way. The pattern writer’s role is to extract implicit knowledge from experts –and convert the knowledge into a short, useful document.

5 Pattern Evolution

The panelists discussed the challenges of updates to previously published narratives. Once a collection of patterns is published, these patterns become difficult to revise, even when a patterns update is needed to add new stories, new context, or new and improved approaches.

Everyone agreed that “out of date research papers” are a lesser problem, not the major crisis that out of date patterns could be. Casper noted that research papers are much lazier about updating their theories than pattern writers – he explained that “I think the half-life of a pattern is probably shorter than the half-life of an academic theory.” Rebecca observed that research papers contain their data and references, but patterns don’t. Rebecca explained her view of the difference: “When I write a research paper, it can stand on its own, it cites other things. When I write patterns, they go out in the wild. They are used by other people, they are misinterpreted by people, they are adapted by people." So you never submit a "recall" of a research paper, you just write a new one. But a pattern needs to evolve to incorporate better techniques and better understanding.

Mary Lynn believed that good pattern writers must try to be humble. A pattern is never "final" and should not be "etched in stone." A pattern must be open to revision by its author or by others.

Rebecca agreed, explaining that many patterns are frequently revised, and that patterns "… cannot stand without a curation and a community. Curation is something that takes time, energy, and passion."

Mary Lynn added that the patterns community has the concept of a "proto-pattern," a trial pattern that needs more validation. She explained that when you call your idea a proto-pattern, you indicate "I think this is true, and I'd like to open it for discussion."

6 The Stories We Tell: The Agile Manifesto

The panel moderator, Steve Fraser (Innoxec), prompted the panelists to discuss the Agile Manifesto [5]: "Where does the Agile Manifesto fit into the stories we tell?" His question triggered several interesting comments, after he reminded panelists that Steve McConnell (Construx) was scheduled for a keynote talk the following day to advocate for revision and modernization of the Agile Manifesto.

Rebecca responded, "The authors [of the Manifesto] aren't going to change it." She added that the Manifesto captured the spirit of what agile was trying to do 20 years ago, but it should not be considered a set of unchanging principles for today. She admired the bravery of the pioneers of Agile because they decided to tell a bold story as a way to gain attention. She said, "How many times do you write a manifesto when you want to do something in research?" She compared the Manifesto authors to members of the early XP community who were gutsy enough to rally around the controversial name "Extreme Programming" coined by Kent Beck to describe a lightweight small-team development process incorporating many practices that we now consider "Agile."

Ken shared his opinion on the Agile Manifesto: "It is a wonderful historical artifact," and he suggested that many are unaware of the Manifesto and its values. Ken thought the most important part of the Manifesto's "story" is its first line: "We are uncovering new and better ways of doing things."

Ademar weighed in about the changing context of the agile world. "Maybe the generality [of the Manifesto] is too general today, we may need to be more specific." Ademar related this to the problem of writing good patterns – patterns that are too generic fail to give useful advice, but patterns that are too specific may age quickly as developers move to new systems and new contexts. To this point, Steve McConnell's XP 2021 keynote gave specific recommendations for updating the Manifesto. Steve indicated several Manifesto principles that should be updated in the light of 20 years of software engineering research and experience.

Mary Lynn thought that "getting your ideas out" for public discussion is the most important lesson from the Manifesto: "The important thing is that people just write."

Mary Lynn gives the Manifesto authors credit for saying what they thought. We should do the same – put out our opinions and let people discuss them.

Casper opined that he was ready to move on from the Agile Manifesto. "I'm interested in what works, what makes the software industry better. We shouldn't start focusing on this like it's a kind of sacred text. It should not be a test of your faith."

7 The Future of Storytelling

Ken believed that social media tools are useful for sharing stories and information. He cited Discord, Twitch, and Twitter as tools to disseminate informal communications. Ken postulated that video, augmented reality, and virtual reality tools may become useful. Ken related how some agile teams record short videos to explain certain architecture decisions or to share other key information with teammates.

Casper observed that video is somewhat imperfect and inefficient. Developers can share information on StackOverflow which could be complemented by video chat. Casper thought that similar sites may emerge for patterns or organizational innovations.

Mary Lynn emphasized that short communications will be effective in the future. "We say that their [developer's] attention span is shorter. I don't know if I believe that. I just think we are being pulled in too many directions." She advocated short videos, short blog entries, and other kinds of short presentations will be more valuable than full research papers and experience reports of 10 pages or more. Short presentations are more likely to be read and discussed, so they will have a greater impact.

Rebecca believed that all forms of storytelling will be relevant in the future. "I think that no form of communication is going to go away. We are just going to add to our storytelling bag of tricks. The challenge we have is picking the right mix."

8 Other Insights from the Panel

Ademar shared the FAIR principles [6] from the Research Communications and e-Scholarship community: knowledge should be Findable, Accessible, Interoperable, and Reusable. E-Scholarship and patterns have communities that are advocates for efficient dissemination of information. Ademar believed that the FAIR principles apply to the creation of good patterns.

Casper asked some probing questions about the pattern writing process. He was curious how a pattern writer could use one or two stories to "abstract the context" of the problem well enough to write a good pattern. Casper asked, "How deep is your understanding?… to me, the patterns look kind of like witchcraft."

Mary Lynn disagreed, explaining that if you could write a pattern from one story, that would be witchcraft. She explained that for years, the patterns community followed the "Rule of Three" – whatever you discovered might not really be a pattern until you hear it three times. She recommended that a pattern author record the sources of the stories that were incorporated into a pattern, because maintaining a pattern is easier if the author can say "this is where I heard this."

The number of references to outside papers and experts should be much lower in experience reports than for research papers, according to Rebecca. "In a story about an agile experience, some people feel that they need to cite all the experts who have written the books." For an experience report, the emphasis should be on what the authors have experienced and learned. "Be very direct about successes or failures, 'aha moments' you might have had… that's what makes an interesting story."

9 Summary

All storytelling formats are useful for sharing experiences and transferring knowledge and technology in the software community.

Ademar recommended trying to write patterns: "It's not harder or easier than a research paper, it's just different."

Ken was quite willing to expand storytelling beyond the three main forms: "I find the practice of writing – whether it's experience reports, research papers, tweets, blog posts, patterns, or anything else – to be a very useful exercise in coordinating my own thoughts, and in helping other people to coordinate and articulate their thoughts."

Casper found that sharing stories, whether in written form or face-to-face over a beverage, can make a positive contribution, even if the stories talk about our problems: "So people don't need to do the same mistakes, and at least we can laugh together."

Mary Lynn and Rebecca both advocated writing as a good "thinking tool," even for people who are ashamed of their own writing style. "You don't have to classify yourself as a writer, just go out there and write," Mary Lynn exhorted. Rebecca was also encouraging: "Not all of us are doing revolutionary things. But you can still seize the moment."

Effective writing will continue to be a means for software professionals to document and share results to continue and advance innovation and technology.

References

1. Johnson, R.E., Beck, K., Booch, G., Cook, W., Gabriel, R., Wirfs-Brock, R.: How to get a paper accepted at OOPSLA (panel). ACM SIGPLAN Notices **28**(10), 429–436 (1993). https://doi.org/10.1145/167962.165934
2. Fraser, S., Mancl, D.: No Silver Bullet: Software Engineering Reloaded. IEEE Softw. **25**(1), 91–94 (2008). https://doi.org/10.1109/MS.2008.14
3. Agile Alliance Experience Reports Initiative website. Available at https://www.agilealliance.org/resources/initiatives/experience-report-program. Accessed 3 Jul. 2021
4. XP 2021 Call for Experience Report Submissions website. Available at https://www.agilealliance.org/xp2021/call-for-submissions/experience-reports. Accessed 3 Jul. 2021
5. Agile Manifesto website. Available at http://agilemanifesto.org. Accessed 3 Jul. 2021
6. Guiding Principles for Findable, Accessible, Interoperable and Re-usable Data Publishing version b1.0. Available at https://www.force11.org/fairprinciples. Accessed 3 Jul. 2021

The Future of Software Engineering: Where Will Machine Learning, Agile, and Virtualization Take Us Next?

Dennis Mancl[1](✉) and Steven D. Fraser[2]

[1] MSWX Software Experts, Bridgewater, NJ 08807, USA
dmancl@acm.org
[2] Innoxec, Santa Clara, CA, USA

Abstract. Software has become the lifeblood of the 21st century, enabling a broad range of commercial, medical, educational, agricultural, and government applications. These applications are designed and deployed through a variety of software best practices. With the onset of the COVID-19 pandemic, developers have embraced virtualization (remote working) and a variety of strategies to manage the complexity of global development on multiple platforms. However, evolving hazards such as network security, algorithm bias, and the combination of careless developers and deliberate attacks continue to be a challenge. An XP2021 panel organized and chaired by Steven Fraser debated the future of software engineering and related topics such education, ethics, and tools. The panel featured Anita Carleton (CMU's SEI), Priya Marsonia (Cognizant), Bertrand Meyer (SIT, Eiffel Software), Landon Noll (Independent Consultant), and Kati Vilkki (Reaktor).

Keywords: Agile · AI · Applications · Collaboration · Education · Machine learning · Professionalism · Remote working · Societal needs · Software engineering

1 Introduction: The Panelists Share Their Views of the Future

Software development has evolved over the past seventy-five years to meet the changing challenges of software product development. Software development has embraced many innovations in technical and business practices: structured programming, waterfall development processes, OO design, automated testing, outsourcing, open source, networking, offshoring, and Agile methods. New programming languages, SDKs (software development toolkits), and code management tools have improved productivity. Today, there is hope that emerging technologies such as machine learning, virtual communication, and remote working tools will improve the reliability and resilience of software systems.

Technology needs to address a crisis of complexity. Yesterday's batch-oriented mainframe systems were relatively simple, but today's software developers face more complexity. Developers work in a global environment targeting multiple platforms while

P. Gregory and P. Kruchten (Eds.): XP 2021 Workshops, LNBIP 426, pp. 222–230, 2021.
https://doi.org/10.1007/978-3-030-88583-0_23

managing ubiquitous networks, petabyte datasets, and emergent hazards. New technology could be useful in this complex environment. For example, some of work of developers might be replaced by automated development and machine learning [1, 2]. In this panel session, the panelists expressed a wide spectrum of opinion on the future.

The panelists focused less on technology and more on the ongoing challenges of the software industry. Software teams everywhere must face issues connected to quality, education, outsourcing, ethics, a push to democratize development, and more connections between computing and non-technical fields. The panelists expressed their anticipation for the benefits of cloud technology, ubiquitous computing, smarter development environments, and systems that leverage natural language processing. The panelists expressed concern about two missing pieces in the software industry. First, there is a need for better-trained software professionals who understand the discipline of product development in light of the challenges of security, safety, and ethical economic viability. Second, industry needs an incentive system that not only rewards innovation – but also penalizes companies that knowingly deliver systems that fail or worse yet cause bodily harm.

1.1 AI and Machine Learning

Two panelists, Anita Carleton (Software Engineering Institute) and Priya Marsonia (Cognizant), embraced the adoption of new technologies, including artificial intelligence and machine learning, to extend the capacity of software developers. Anita is Director of SEI's Software Solutions Division, and Priya is a Senior Client Partner at Cognizant.

Anita anticipated that AI will support the development of software systems. She expected that the combination of human and machine intelligence will enable the software industry to keep ahead of a dynamically changing environment. AI-based development tools will help meet increasing quality requirements.

Priya was convinced that we will see more natural language programming and better tool support for developers. She explained, "In the agile world, I see the infrastructure as helping us more," with humans collaborating with intelligent agents and machine-identified "digital twins." A future software development environment might be able to analyze the programming styles of team members, then assign pair programming partners based on similarity of programming style. In an agile environment of the future, a developer may pair program with an intelligent agent-selected partner, or even an AI instead of a human partner.

Priya anticipated that another fruitful area for automation is to provide support for large multi-team projects. Systems could scan enormous volumes of complex subsystems, assessing code similarities and offering suggestions of where teams may need to collaborate. Inter-team communication can avoid duplication, discover useful lessons from the past, or improve the testing process. Priya added, "The whole notion of getting suggestions and spotting trends would be embedded in our environment, and we would have to do fewer and fewer rudimentary operations."

1.2 Conventional Technology with Better Failure Analysis

Bertrand Meyer (Schaffhausen Institute of Technology and Eiffel Software), agreed with the positive assessment of the future, but believed that more conventional technology and processes will continue to advance.

Bertrand asserted that there are reasons to be proud of what software has done (including virtual conferences). "Imagine the depth of the stack just to have this [this panel session]." But we have also seen big failures. He referred to the 7-hour outage of emergency services telephone service in France in the first week of June 2021 [3]. "This was a software bug. Why don't we have the same kind of analysis we have for airplane accidents?" One of software engineering's biggest challenge is "correctness" – and a public airing of significant bugs will help developers to understand the root causes of future software failures.

1.3 Need to Address Software Failures

Landon Noll (Landon Noll and Associates) was more pessimistic. He warned that we are already in a software crisis, and we need major changes to improve the education and management of software professionals to increase future "readiness."

Landon explained his views – and why he believes that skills and best practices are not improving. "So many companies and developers use the term software engineer – yet [they] lack any formal training or certification to understand even basic engineering principles." This is a view shared by software engineering thought leaders, such as David Parnas and Mary Shaw [4][5][6]. Landon complained that companies in non-critical industries don't care, and he blamed the limits on legal liability for software that fails – and users accustomed to accepting poor quality software. Landon claimed that we need to address these skill deficits and fix these economic incentives to make progress.

1.4 Less Outsourcing in Data-Driven Industries

Kati Vilkki (Reaktor) shared her consulting experience in digital transformations based on her years as an agile technology leader at Nokia. Kati observed that many banks and insurance companies have gradually "outsourced" many of their information technology functions. At the time, company executives felt that IT was not part of the company's core business. But today, the pendulum is swinging the other way. These companies have found that their businesses have evolved and that "data is king." Kati reported: "I see them desperately trying to in-source, to hire software developers."

2 Democratization of the Software Industry?

Priya noted that there is an increased "democratization" of software development, with the rise of many new low-code or no-code environments that can be used by non-programmers to build applications. She talked about Cognizant's think tank "Center for the Future of Work," which was launched to study the influences of globalization, virtualization, AI, and the cloud [7]. That future for software engineering, according to

Priya, may be “founded on natural and spoken language, where we don’t have to be as conversant in an arcane coding language, or the vagaries of a very specific environment that’s complex to manipulate.”

Anita added her observations on the evolution of the software industry. She described significant advances in the past 30 years of software development, such as a better focus on architecture, improved practices and tools, agile processes, and “DevSecOps” which have combined to make product development better.

Landon voiced concerns. He noted that, “Agile practices have a potential for helping,” but observed that developers are often attracted by the glitter of new programming languages. He would prefer developers to spend less time on the “language of the month” or “current design fads” and more on engineering principles, best practices, and good algorithm design.

3 Can AI and Machine Learning Help?

The panel was split on the question of the potential effectiveness of AI-centric tools.

In 2020, Anita was one of the guest editors of the July/August 2020 issue of *IEEE Computer* [8], which contained several articles that explored future interactions between AI and software engineering. Anita expected that AI will be part of “societal-scale systems.” However, she worried that we still aren’t sure where AI research will take us. Priya believed that future programming infrastructure will incorporate machine learning technology. But Bertrand was skeptical about the potential impact of machine learning on software development. He explained that the development of correct programs requires logic and reasoning, not heuristic search. “Machine learning works on statistical principles. Program correctness works on a different kind of mathematics – which is logic. It’s not so easy to see how we can apply statistical techniques. What does it mean to say the programs in the world are going to be 2% more correct?”.

4 Companies May Need to Do More In-Sourcing

Kati worried about the consequences of “out-sourcing followed by in-sourcing.” When companies outsourced most of their development, many of them were seeking to reduce costs, so they collected many applications from different vendors. “I don’t think it’s a wonder that the whole thing is a mess.” The process of in-sourcing and hiring software developers is difficult and costly.

Kati shared her experience in Finland – developers are expensive, they can choose where they want to work, and the companies do not have much experience leading and managing software projects. Some accommodation is needed on all sides. “For this to work,” according to Kati, business specialists need to know more about software and software people need to know more about business. She mentioned the increased demand for her consulting services, where leadership teams frequently request her introductory seminar course on software management.

5 Do We Need to Reform the Software Industry?

Three areas of evolutionary or revolutionary change were on the minds of the panelists: reform of the industry's commercial incentives (including product liability), the future of free and open source software, and general improvements in software engineering education.

5.1 The Software Industry and Commercial Incentives

Landon believed the failure of modern software engineering is rooted in commercial incentives. He called for reform: a specific list of actions to change the economic model for software development to reduce company incentives to ship software systems and applications that crash. Landon's list:

- Invalidate EULAs (End User License Agreements) for software or software-based services that attempt to avoid responsibility for the software and services they sell
- Penalize companies with triple damages if they willfully disregard software best practices, lack adequate testing, ignore reported defects, or employ unqualified software developers
- Require certification and continuing education for software developers
- Empower regulatory panels to study software failures and make recommendations e.g., patterned after those that reviewed the Challenger or Titanic incidents
- Identify and adopt "best practices" widely, including for non-critical software systems like games

Other panelists pushed back on these suggestions.

Anita claimed that the software industry has improved steadily and it will continue to advance in many areas, such as better instrumentation of the development process. She went on, "I don't think tools are the only answer. But humans, tools, and AI, working together to address issues of scalability, composability, and complexity" will continue to make progress.

Priya was also more positive about the software industry than Landon. She recognized potential quality issues with the "democratization" of software engineering. While systems may be designed and built by experts, some applications may be developed by end users – so we must be aware of the relative quality differences. In the end, ongoing abstraction and the continuation of original agile principles will mean more people can solve problems using software principles appropriate to their own level of expertise.

Landon was critical of the "fail hard, fail fast" culture often found in the software industry, noting that this emphasis on failure has been an excuse for delivering poor quality software. It will be tragic if the "fail hard, fail fast and throw it out there and see what happens" culture is pervasive, and influences negatively (life threatening results) critical systems such as medical, avionics, and security systems.

Bertrand defended "fail fast" by explaining the agile thinking behind it. "When they say 'fail fast,' it's not that it is OK to fail. It's a different approach to program verification." He explained that agile development emphasizes frequent testing: "Their way of getting things correct is to test it all the time." Bertrand preferred other approaches like careful up-front design and programming by contract, but he insisted that XP and other agile methods can build quality software with an iterative approach combined with a rigorous testing process: “I’m not ready to cast stones at the agile people… It’s not my approach, but it’s completely reasonable, and it is possible to combine it will other approaches.” Bertrand referred the audience to his recent book (*Agile: The Good, the Hype, and the Ugly*) for more details.

Kati explained that the software industry will need to address future issues that cannot be solved within software engineering alone. “We need to have a much better connection to psychology, sociology, neuroscience, environmental sciences, and so on.” In the past, when UX (user experience) was a new concept, “it brought a whole new perspective to many software developers.” Kati expressed that the broader view is critical: “to widen our horizons, think about the impact of software, and make conscious choices.” She explained, “I’m all for good engineering practices and teaching that as a science. But I don’t think it’s enough.”

5.2 The Future of Open Software

Bertrand questioned whether free software and open source software will continue as a major source of future innovation. He knew that criticism of the open source economic model is not a popular point of view. He warned that if you are too critical, “you appear to be a horrible representative of the powers of mercantile capitalism.” He admitted that the world has gained a lot from open source, but he pointed out that the experience of other engineering fields is unanimous: “there is no example of an engineering discipline which has gotten better in a context where there was no money to be made.” Bertrand was willing to consider open source to be “a different business model” that that is useful in some contexts, but the software engineering community needs to be realistic about its limitations.

Some economists view “free” software and services as a “barter” system – free products in exchange for user data. As a recent Harvard Business Review article [9] explained, “The business strategy of companies such as Facebook, Google, and numerous others is partly an exchange that does not entail money: Consumer data is being collected in exchange for the provision of internet services, just as berries might be swapped for meat.” The economics of open source may evolve along a similar path to the economics of social media websites and “free” online services such as Dropbox and Google Docs.

5.3 The Future of Software Engineering Technology and Education

Panelists suggested topics for inclusion in software engineering education and future software engineering technologies.

Landon advocated an approach to focus developers on building applications with an attention to correctness and reliability. “You have to start with the simplest cases [in education and industry]: trivial ‘Hello World’ programs, games, social media, and things that are not critical.” In computer science courses, it is better “to have assignments graded on whether they put in the appropriate amount of documentation and testing, not just on whether the program worked.” Landon advocated assessing assignments based on both development process and operational function.

Kati believed that ethical considerations are a fundamental consideration, particularly in AI applications and surveillance systems. She noted that society is exceptionally vulnerable since so much of our life is heavily reliant on software. “We need to start to think about the impact of what we do [with software] to others.”

Anita mentioned a study launched by SEI last year to anticipate the future of software engineering. That study, the National Agenda for Software Engineering Research & Development, proposed research focus areas and collected ideas on the importance of ethics in software. Other key observations included increases in automation, scalability, evolvability, and rapid deployment, as well as more applications of AI (such as AI-augmented software development).

Priya believed that the future of software engineering will need to combine technological advances with social components – technologies such as AI/ML and virtualization will assist in the democratization of development, build systems that incorporate empathy, and create software that embodies the characteristics of its users and makers. Priya saw this as a natural evolution of the path software engineering was already on.

Bertrand reflected on four influential technology areas that have been making their mark on software engineering over the last 30 years: object-oriented programming, open source, agile development, and cloud-related technologies (which includes microservices and DevOps). Bertrand saw the cloud-related technologies as positive so far (“it’s recent and the jury is still out”), but these techniques are changing how we do software engineering. The technologies cross some boundaries, because cloud, microservices, and DevOps are a blend of software engineering, networking, and systems administration, and they are likely to trigger big changes in application development. Bertrand’s view was that “traditional software engineering wisdom, principles, and modes of reasoning about the world do not completely transpose to that new world.”

6 Summary: Goals for the Future

The panelists shared their disagreements about the future of software engineering, but they agreed that technology, education, and markets will bring new challenges. The software community will find new ways to use AI, machine learning, remote working, cloud technology, natural language processing, and new software engineering tools to meet those new challenges.

Anita: The future will bring us smart automation, AI-inspired automation, evolving systems, composability and scalability of systems, and the architecture of new types of systems.

Priya: We will have a more inclusive software engineering ecosystem where everyone contributes. Building and integrating new software will be interdisciplinary, it will be ubiquitous, and it will be applied at different skill levels with varying ranges of expertise.

Landon: How the software industry integrates with society is key. We need continuing education – just because you are a software developer doesn’t mean you can stop learning. Practitioners will need to have certification and recertification – similar to other safety-critical medical and engineering fields.

Bertrand: Developers need make the right choices as they develop software technologies. Many choices of technology at all levels of the stack are influenced by extraneous criteria: company policies, *a priori* tool selection, the color of the marketing brochure, or whatever.

Kati: Everybody needs to understand something about software development fundamentals, and about "good" software development. Fundamentals need to be part of every single curriculum at the university, especially for future leaders and managers.

Steve Fraser, the panel impresario, offered a pointer to a 2006 OOPSLA panel that had explored the "Future of Agile" [10]. Steve concluded the panel with the observation: "We need to remember the past in order to forge the future."

References

1. Ré, C.: Software 2.0 and Snorkel: beyond hand-labeled data. In: KDD 2018 ACM International Conference on Knowledge Discovery & Data Mining, p. 2876. https://doi.org/10.1145/3219819.3219937 (2018)
2. Karpathy, A.: Software 2.0. https://karpathy.medium.com/software-2-0-a64152b37c35. Accessed 3 Jul. 2021 (2017)
3. France emergency service number disrupted after network outage, 3 Jun 2021. https://www.bbc.com/news/world-europe-57341526. Accessed 3 July 2021
4. Parnas, D.L.: Software engineering: a profession in waiting. IEEE Comput. **54**(5), 62–64 (2021). https://doi.org/10.1109/MC.2021.3057685
5. Shaw, M.: Research toward an engineering discipline of software. In: Proceedings of FoSER 2010: FSE/SDP Workshop on Future of Software Engineering Research, pp. 337–341. https://doi.org/10.1145/1882362.1882431 (2010)
6. Shaw, M.: Progress toward an engineering discipline of software. Video of a talk for the SATURN 2015 conference, https://www.youtube.com/watch?v=S03bsjs2YnQ. Accessed 3 July 2021 (2015)
7. Cognizant: The future of work website. https://www.cognizant.com/future-of-work. Accessed 3 July 2021 (2021)
8. Carleton, A.D., Harper, E., Menzies, T., Xie, T., Eldh, S., Lyu, M.: The AI effect: working at the intersection of AI and SE. IEEE Softw. **37**(4), 26–35 (2020). https://doi.org/10.1109/MS.2020.2987666
9. Tett, G.: The data economy Is a Barter economy. Harvard Business Review, 6 Jul 2021 (2021)
10. Fraser, S., Rising, L., Ambler, S., Cockburn, A., Eckstein, J., Hussman, D., Miller, R., Striebeck, M., Thomas, D.: A fishbowl with piranhas: coalescence, convergence, or divergence? The future of Agile software development practices. In: Companion to OOPSLA '06, pp. 937–939. https://doi.org/10.1145/1176617.1176750 (2006)

Author Index

Berntzen, Marthe 52

Cherrueau, Ronan-Alexandre 176
Copei, Sebastian 186

Das, Suddhasvatta 3
Delavergne, Marie 176
Dobrigkeit, Franziska 201

Evertse, Robert 96

Fraser, Steven D. 215, 222

Gary, Kevin 3
Gustavsson, Tomas 12, 33

Heimann, Christian 42

Jansen, Slinger 96, 125

Kettunen, Petri 33
Kottke, Mario 42

Laanti, Maarit 33
Lebre, Adrien 176
Lencz, Abel 96

Mancl, Dennis 215, 222
Männistö, Tomi 33
Matthies, Christoph 201
Melegati, Jorge 155
Mikalsen, Marius 135
Mikkonen, Tommi 33
Moe, Nils Brede 23, 52, 135, 145
Ochoa-Zambrano, Juan 69

Onyango, Samuel 125

Pajak, Philipp 201
Petrik, Dimitri 115
Philip, Matthew 207
Poth, Alexander 42

Riel, Andreas 42
Rygh, Nina 135

Sachweh, Sabine 167
Saltan, Andrey 87
Scholten, Joram 125
Šinik, Tea 96
Šmite, Darja 23
Smolander, Kari 87, 106
Soussi, Lamia 96
Sporsem, Tor 135, 145
Springer, Virginia 115
Steenvoorden, Emilie 125
Stray, Viktoria 23, 52

Teusner, Ralf 201
Thirion, Yoan 207
Tjernsten, Andreas 33
Tkalich, Anastasiia 135, 145
Trebbau, Simon 167

Ulfsnes, Rasmus 23
Uwadi, Maduka C. 195

Vedal, Henrik 52
Vuolasto, Jaakko 106

Wickert, Manuel 186
Wizenty, Philip 167

Zündorf, Albert 186