Managing Knowledge Resources in Agile Software Development

Raquel Ouriques

Department of Software Engineering
Blekinge Institute of Technology

Doctoral Dissertation Series no. 2023:14

Blekinge Institute of Technology
Doctoral Dissertation Series No. 2023:14
ISSN 1653-2090
ISBN 978-91-7295-467-0

Managing Knowledge Resources in Agile Software Development

Raquel Ouriques

DOCTORAL DISSERTATION
for the degree of Doctor of Philosophy at Blekinge Institute of Technology to be publicly defended on October 5th at 14:00 in J1360, Campus Karlskrona
Supervisors
Daniel Mendez, Fabian Fagerholm and Tony Gorschek
Faculty opponent
Rebekka Wohlrab

Abstract

Context: Many software companies adopt Agile Software Development (ASD) principles through various methods, aiming to respond rapidly to market changes or internal transformations. As agile methods prioritise intense communication between people over documentation to bring more flexibility and readiness to welcome changes, the pressure on how knowledge is shared and applied increases. Many knowledge resources remain intangible in these contexts, requiring lots of effort to understand what should remain tacit and what should be captured explicitly as artefacts.

Objective: This thesis aims to contribute to a better understanding of knowledge resources in agile software project environments and provide guidance on effectively managing them.

Method: We follow mostly a qualitative approach. We adhere to social constructivism research, which notes that social phenomena undergo constant changes and are affected by human interaction. As qualitative and quantitative methods of investigation, we utilised literature reviews, grounded theory, survey and a case study.

Results: We synthesised evidence from the literature to show the proportions of knowledge management practices utilised in ASD environments and the knowledge process they focus on. Through a grounded theory study, we identified Knowledge-based Resources (KBRs) that support changes in agile environments in the Knowledge-push theory. In this study, we identified inefficiencies in converting KBRs into Property-based Resources (PBRs). This evidence led us to a case study in which we investigated the causes and effects of trust in Boundary Artefacts (BAs), a specialisation of PBRs. The results have contributed to understanding the favourable factors that make stakeholders feel confident in utilising BAs and pointed to the implication of decreased trust in software projects. Such negative implications can be mitigated by applying our developed and validated guideline that supports the creation of BAs in software engineering, which was perceived as being able to increase the trustworthiness of BAs. Lastly, we gathered the evidence that we collected through this doctoral journey and offered a simplified discussion about knowledge resources in an agile context.

Conclusions: We clarify the concept of KBRs, identify them, and explain how they support changes in agile contexts. In this process, we uncover the inefficiencies in converting KBRs into PBRs and support their management in software projects. For example, (i) understanding how trust aspects such as reliability, predictability, and functionality affect practitioners' confidence in BAs, (ii) providing a structured guideline that helps practitioners create BAs, (iii) incorporating more formal practices to manage BAs that do not necessarily abandon agile flexibility to deal with changes.

Keywords: Knowledge Management, Agile Software Development, Knowledge Resource, Boundary Artefacts

Blekinge Institute of Technology
Doctoral Dissertation Series No. 2023:14

Managing Knowledge Resources in Agile Software Development

Raquel Ouriques

Doctoral Dissertation in Software Engineering

Department of Software Engineering
Blekinge Institute of Technology
SWEDEN

Blekinge Institute of Technology
Department of Software Engineering

Blekinge Institute of Technology Doctoral Dissertation Series No. 2023:14
ISBN 978-91-7295-467-0
ISSN 1653-2090
urn:nbn:se:bth-25299

Printed in Sweden by Media-Tryck, Lund University, Lund 2023

“Emancipate yourselves from mental slavery.”
“None but ourselves can free our minds.”
- Redemption song.

Abstract

Context: Many software companies adopt Agile Software Development (ASD) principles through various methods, aiming to respond rapidly to market changes or internal transformations. As agile methods prioritise intense communication between people over documentation to bring more flexibility and readiness to welcome changes, the pressure on how knowledge is shared and applied increases. Many knowledge resources remain intangible in these contexts, requiring lots of effort to understand what should remain tacit and what should be captured explicitly as artefacts.

Objective: This thesis aims to contribute to a better understanding of knowledge resources in agile software project environments and provide guidance on effectively managing them.

Method: We follow mostly a qualitative approach. We adhere to social constructivism research, which notes that social phenomena undergo constant changes and are affected by human interaction. As qualitative and quantitative methods of investigation, we utilised literature reviews, grounded theory, survey and a case study.

Results: We synthesised evidence from the literature to show the proportions of knowledge management practices utilised in ASD environments and the knowledge process they focus on. Through a grounded theory study, we identified Knowledge-based Resources (KBRs) that support changes in agile environments in the Knowledge-push theory. In this study, we identified inefficiencies in converting KBRs into Property-based Resources (PBRs). This evidence led us to a case study in which we investigated the causes and effects of trust in Boundary Artefacts (BAs), a specialisation of PBRs. The results have contributed to understanding the favourable factors that make stakeholders feel confident in utilising BAs and pointed to the implication of decreased trust in software projects. Such negative implications can be mitigated by applying our developed and validated guideline that supports the creation of BAs in software engineering, which was perceived as being able to increase the trustworthiness of BAs. Lastly, we gathered the evidence that we collected through this doctoral journey and offered a simplified discussion about knowledge resources in an agile context.

Conclusions: We clarify the concept of KBRs, identify them, and explain how they support changes in agile contexts. In this process, we uncover the inefficiencies in converting KBRs into PBRs and support their management in software projects. For example, (i) understanding how trust aspects such as reliability, predictability, and functionality affect practitioners' confidence in BAs, (ii) providing a structured guideline that helps practitioners create BAs, (iii) incorporating more formal practices to manage BAs that do not necessarily abandon agile flexibility to deal with changes.

Keywords: *Knowledge Management, Agile Software Development, Knowledge Resource, Boundary Artefacts*

Acknowledgements

I express my gratitude to my supervisors, Daniel Mendez, Fabian Fagerholm and Tony Gorsheck, for all their insightful guidance and feedback during this PhD journey. I am also grateful to Krzysztof Wnuk and Richard Berntsson Svensson for being part of the supervision team previously. Additionally, this endeavour would not have been possible without the generous support from KKS Foundation, which financed my research.

I want to extend my sincere thanks to the companies Ericsson Karlskrona and Axis Communications and their employees for making possible data collection and participating in the workshops, especially Baldvin Gislason Bern, for all the insightful discussions.

I am also grateful to my colleagues at SERL Sweden for their help and moral support. Thanks should also go to colleagues in other departments who impacted and inspired me.

Lastly, I would be remiss in not mentioning my family, my spouse Felipe Ouriques, my parents, and son. Their belief in me has kept my spirits and motivation high during this process.

Overview of Papers

Papers in this Thesis

- **Chapter 2: Raquel Ouriques**, Krzysztof Wnuk, Tony Gorschek, and Richard Berntsson Svensson. "Knowledge Management Strategies and Processes in Agile Software Development: A Systematic Literature Review" *International Journal of Software Engineering and Knowledge Engineering*, 29(3): 345–380, (2019).

- **Chapter 3: Raquel Ouriques**, Krzysztof Wnuk, Tony Gorschek, and Richard Berntsson Svensson. "The role of knowledge-based resources in Agile Software Development contexts" *Journal of Systems and Software*, 197, 111572, (2023).

- **Chapter 4: Raquel Ouriques**, Fabian Fagerholm, Daniel Mendez, and Baldvin G. Bern. "An investigation of causes and effects of trust in Boundary Artefacts". *Information and Software Technology Journal*, 158, 107170, (2023).

- **Chapter 5: Raquel Ouriques**, Krzysztof Wnuk, Richard Berntsson-Svensson and Tony Gorschek. "Thinking Strategically About Knowledge Management in Agile Software Development." *In Kuhrmann M. et al. (eds) Product-Focused Software Process Improvement. PROFES 2018. Lecture Notes in Computer Science,* LNCS 11271, 389–395, (2018).

- **Chapter 6: Raquel Ouriques**, Fabian Fagerholm, Daniel Mendez, Tony Gorschek, and Baldvin G. Bern. "Preliminary Guideline for Creating Boundary Artefacts in Software Engineering". *Submitted to Information and Software Technology Journal (Under review)*, IEEE, (2023).

- **Chapter 7: Raquel Ouriques**, Tony Gorschek, Daniel Mendez, Fabian Fagerholm. "Connecting the Dots of Knowledge in Agile Software Development". *Submitted to Communications of the ACM (CACM) (Under review)*, (2023).

Contribution Statement

Raquel Ouriques is the first and leading author of all papers reported in this thesis. The contributions of each author are described through the Contributor Roles Taxonomy (CRediT) [109].

Chapter 2

- Raquel Ouriques: conceptualisation, formal analysis, investigation, visualisation, writing - original draft, writing review and editing. Krzysztof Wnuk: conceptualisation, data curation, visualisation, writing - review and editing. Tony Gorschek: funding acquisition, visualisation, writing - review and editing. Richard Berntsson Svensson: visualisation, writing - review and editing.

Chapter 3

- Raquel Ouriques: conceptualisation, formal analysis, investigation, visualisation, writing - original draft, writing review and editing. Krzysztof Wnuk: investigation, visualisation, writting - review and editing. Tony Gorschek: funding acquisition, writing - review and editing. Richard Berntsson Svensson: writing - review and editing.

Chapter 4

- Raquel Ouriques: conceptualisation, data curation, formal analysis, investigation, visualisation, writing - original draft, writing - review and editing. Fabian Fagerholm: Conceptualisation, visualisation, writing - review and editing. Daniel Mendez: Conceptualisation, visualisation, writing - review and editing. Baldvin Gislason Bern: project administration, data curation, writing - review and editing.

Chapter 5

- Raquel Ouriques: conceptualisation, visualisation, writing - original draft, writing - review and editing. Krzysztof Wnuk: writing - review and editing. Tony Gorschek: funding acquisition, writing - review and editing. Richard Berntsson Svensson: writing - review and editing.

Chapter 6

- Raquel Ouriques: conceptualisation, project administration, data curation, formal analysis, investigation, visualisation, writing - original draft, writing - review and editing. Fabian Fagerholm: Daniel Mendez: Tony Gorschek: funding acquisition, writing - review and editing. Baldvin Gislason Bern: project administration, writing - review and editing.

Chapter 7

- Raquel Ouriques: conceptualisation, writing - original draft, writing - review and editing. Tony Gorschek: funding acquisition, conceptualisation, writing - original draft, writing - review and editing. Daniel Mendez: writing - review and editing. Fabian Fagerholm: writing - review and editing.

Other Papers not in this Thesis

- **Paper 1:** Panagiota Chatzipetrou, Raquel Ouriques, and Javier Gonzalez-Huerta. “Approaching the Relative Estimation Concept with Planning Poker". *In Proceedings of the 7th Computer Science Education Research Conference (CSERC '18)*, ACM, New York, NY, USA, 21-25, 2018.

- **Paper 2: Raquel Ouriques**, Ricardo Britto, Krzysztof Wnuk, João Felipe Ouriques, and Tony Gorschek. "A method to evaluate knowledge resources in agile software development." In Proceedings of the 13th *ACM/IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM)*, 2019.

Funding

This work was supported by the KKS foundation through the S.E.R.T. Research Profile at Blekinge Institute of Technology.

Contents

List of Abbreviations

Abbreviation	Definition
TD	Technical Debt
ASD	Agile Software Development
BA	Boundary Artefact
DSDM	Dynamic Software Development Method
KA	Knowledge Application
KC	Knowledge Creation
KM	Knowledge Management
KP	Knowledge Process
KBR	Knowledge-based Resource
KS	Knowledge Storage
KT	Knowledge Transfer
KMS-ASD	Knowledge Management Strategy in Agile Software Development
PBR	Property-based Resource
SLR	Systematic Literature Review
XP	Extreme Programming

List of Figures

List of Tables

Chapter 1

Introduction

Many software companies adopt Agile Software Development (ASD) through an assortment of principles and practices. They do so because they operate in a fast-changing environment, and traditional (plan-driven) approaches are not flexible enough to let the organisations rapidly respond to changes. Agile methods became notorious as they promised shorter cycles and better adaptability in turbulent environments.

Agile principles advocate that teams should focus on frequent communication between their members over formalised documentation to bring more flexibility and readiness when welcoming changes [39]. In addition, team communication capability, collaborative skills, and creativity are all vital success factors in achieving the promised flexibility [148].

In such a collaborative environment, communication facilitates knowledge sharing, which becomes critical as it has great potential to help embrace changes. That is only possible because people share what they know with others, and combining knowledge can result in a competitive advantage, e.g. innovation is one way.

Maximising produced value through efficient resource allocation is one of the keys to sustaining companies' competitive strategies [11]. As an intangible resource, knowledge is difficult to reproduce and manage, requiring lots of effort to understand what should remain tacit and what should be captured explicitly in the form of artefacts.

When kept in its intangible form, employees' knowledge, which we call Knowledge-based Resources (KBRs), refers to specific technical, creative, and collaborative skills. As KBRs are stored in documents or transformed into products, they become Property-based Resources (PBRs) because companies now own them by legal rights and can generate profit over them. They can be transformed into economic value when revealed, stored and applied, i.e., prototypes, architectural descriptions, requirements documentation, and artefacts in general [134].

In contexts where agile principles are prioritised, many of these KBRs remain intangible, which poses challenges to resource management because people hold the knowledge and can only be utilised by the company for profit purposes as it is externalised into, for example, artefacts.

Two strategies to manage knowledge resources are often reported in the literature: personalisation and codification [78]. Personalisation focuses on practices that stimulate human interaction for KBRs sharing and creation in, for example, communities of practices. In codification strategies, companies extract knowledge from employees and store it in artefacts or other formats, transforming knowledge into PBRs.

In research, knowledge in agile environments has received attention since 2003, and studies on this topic have increased remarkably since 2010 [148]. Researchers started reporting mainly case studies and exploratory studies, most of which focused on personalisation strategies. The studies provided a good comprehension of how software companies understand knowledge resources and use strategies to manage them in agile contexts. However, we also observed that there are still open issues involving the clarification of KBRs in software development that requires some attention [134]. Moreover, although codification strategies have been reported as a minor strategy compared to personalisation, the inefficiencies originating in the application of such strategy can be harmful in many ways to software projects. This issue has not been fully investigated. We observed two main gaps in the scientific literature, which we address in this thesis:

First, **evidence and insights on the materialisation of knowledge resources in software project environments is scarce (gap 1)**. While most literature focuses mainly on knowledge-sharing processes due to extensive focus on personalisation strategies, other processes have been neglected, such as creation, application and storage. Moreover, the identification of KBRs in agile environments remains unclear as to how they are utilised to perform constant changes.

Second, **the research literature offers little guidance on effectively managing knowledge resources in software project environments (gap 2)**. Several studies describe practices to support knowledge sharing in codification or personalisation strategies. However, these practices are often isolated, and it is unclear how to effectively utilise them when referring to codification strategies. Moreover, a deep understanding of why codification often results in inefficiencies and an exploration of means to address them is missing.

Structure

This thesis is structured in two main parts. The first part, which we refer to in this introductory chapter, introduces the topic by motivating it and presenting the research gaps. In sequence, we present the conceptual background underpinning this thesis in Section 1.1. In sequence, we present the research methodology adopted to answer our research questions in section 1.2. In Section 1.3, we summarise the results of each investigation, and discuss them in Section 1.4. Contributions to literature, the implication for practice and future research are detailed in Section 1.5. Lastly, we conclude this first part in Section 1.6.

The second part of this thesis comprehends the remaining chapters of this thesis 2, 3, 4, 5, 6, and 7, which provide an overview of the six papers included in this thesis.

1.1 Background

This section contextualises our research topic by offering a brief overview of the characteristics of ASD. We also provide a number of definitions of the concept of 'knowledge' that we have adopted in this study. Furthermore, this section provides an overview of the theoretical foundations.

1.1.1 Agile Software Development

Traditional software development processes struggle to offer many software companies the required flexibility for making early changes in software projects [39]. Promptly responding to changes is the prevalent idea of ASD that had its principles disseminated in 2002 [14]. ASD principles arise as a contradiction to the traditional software development which is inspired by the project cycle and guided by tasks and documents [137]. Instead of extensive documentation to guide the product development, the ASD relies on the team capability of communicating, collaboration, and being creative [39].

Successfully adopting ASD requires several adaptations, since it highly depends on the context that is applied. Contextual factors such as system size, customer demand for software, team expertise, and highly constrained quality attributes requires evaluation in agile adoptions [104]. A flexible organizational structure is also required for agile software development, where collaboration takes place.

Agile software development methods have been adopted in small collocated teams and also in large companies that have distributed teams [32, 122, 151]. Studies had already reported success on this adoption, although several challenges remain. Those challenges are related to team coordination, requirements definition, communication and knowledge sharing [112, 152].

The ability to deal with constant change is the central idea of ASD. Given this, we note that knowledge plays an essential role in this regard [161]. While ASD prioritises informal communication (due to the flexibility and reduced demands for documentation in ASD), it also largely relies on knowledge exchanged between people [39].

1.1.2 Knowledge conceptualisation

The nature of knowledge originates in the philosophy of science (epistemology) and has been discussed extensively among philosophers such as Plato, Aristoteles, Nietzche, and Polanyi [73,84]. They focus primarily on describing the theory of knowledge and comparing the different perspectives, such as opinion versus belief [84].

From the knowledge management perspective, knowledge has been mostly seen from a practical point of view, where its utility is more important as it can provide companies with a competitive advantage. We adhere to this perspective and consider a somewhat straightforward definition of the concept of 'knowledge' as "a collection of information that provides guidance [that is] based on individual cognitive processes" [3, 139].

The two most frequently cited types of knowledge are widely debated, namely *explicit knowledge* and *tacit knowledge*. Explicit knowledge is systematised in formal language (for example, in manuals, specifications, and other documents) and any documented format (e.g., graphics, designs, drawings), whilst tacit knowledge is highly personal since it is rooted in people's actions, values, and routines. Consequently, it is difficult to formalise and systematise tacit knowledge in a formal language [141].

Knowledge Management (KM) strategy is the effort to formulate plans to actively manage knowledge [182]. The strategies can be codification and/or personalization strategies, which are rooted on the company's corporate strategy [23]. Codification involves the storage of Knowledge Management Strategies and Processes in Agile Software Development knowledge into databases for further use, e.g. wiki-based tools [92], while personalization focus on human interaction to communicate knowledge (e.g. communities of practice) [78]. The KM strategies promote the set of knowledge processes that represent individual's cognitive, social practices and the culture that shapes knowledge in organizations [3, 46, 141]. There processes are:

- Creation - the combination of tacit knowledge for new knowledge generation;
- Codification/storage - a means for systematising and storing relevant knowledge as useful information;
- Transfer/sharing - the movement of both types of knowledge between people or groups; and
- Application - the use of knowledge to generate value.

The combination of these processes has, as its final product, codified knowledge to be shared and applied by co-workers. The first stage is to partially externalise tacit knowledge, which will be combined with other knowledge (creation process), stored, transferred, and, lastly, applied for-profit purposes [139].

Knowledge resources

According to a resource-based view, a company's competitive strategy way significantly dependent on how the company's management team maximises the produced value by allocating the company's resources. These resources are idiosyncratic and thus provide an inevitable heterogeneity to the company, as it operates within a market [11].

Resources that companies use can be property-based or knowledge-based. Property-based resources are owned by a company and are protected by legal rights, for example, by contracts and patents. Knowledge-based resources (KBRs) *is defined as the employees' intangible knowledge that is valuable to a company's competitive advantage.* KBRs appear as specific skills, including technical, creative, and collaborative skills, which, in turn, are immediately relevant to the integration and coordination of multidisciplinary teamwork [134].

The KBRs receive more attention when they are involved in activities that require human interaction for production, inside and outside the company. For example, this may include team and customer collaboration [9, 169]. KBRs are also often used to develop property-based resources, for example, software documentation and process descriptions.

An intangible portion of co-workers' knowledge (KBRs) is revealed when stored and applied through, e.g., prototypes, architectural descriptions, requirements, and other artefacts. As this knowledge is typically stored in artefacts, they become PBRs as companies own them and can generate profit utilising them [134].

However, a large proportion of these KBRs remain intangible, and are, thus, not realised into PBRs. There might be several reasons for this. The domain of the company might be moving too fast to create PBRs and PBRs might become outdated too fast to warrant the formalisation investment. Another possible scenario is the complexity of generating PBRs out of KBRs that are usable and useful for others. Knowledge in terms of KBRs are, thus, a reality that companies live with, and they strongly support software development companies. This thesis makes use of knowledge management and resource concepts to support the understanding of knowledge resources in agile software project environments and improve their management.

1.1.3 Boundary artefacts

The concept of Boundary Artefacts (BA) has its roots in the field of ecology [111, 174, 175]. The term "boundary object" was initially introduced in 1989 in a study conducted for the Museum of Vertebrate Zoology at the University of California [175]. The authors of the study defined it as "objects that possess sufficient flexibility to accommodate the specific requirements and limitations of multiple parties involved, while also retaining a consistent identity across different locations." In this thesis, we explore BAs as a specialisation of KBRs.

In the literature, two commonly used terms are "boundary objects" and "boundary artefacts." Since our focus is on the discipline of software engineering, we will use the term "boundary artefact" (BA) as it aligns closely with the field. While we recommend referring to Mendez et al. [132] for a more detailed discussion and definition of the term, in simple terms, an artefact refers to a tangible or intangible product generated, modified, or utilized in a series of tasks that hold value for a particular role. Examples of artefacts include code, test cases, or requirements documentation.

Boundaries within organizations do not always act as strict limits; instead, they are often permeable. Rather than serving as barriers for information, boundaries facilitate transitions where individuals on one side perceive information from their perspective, while those on the other side perceive it from a different viewpoint. Interestingly, both groups may be looking at the same information or working towards the same goals. For instance, a use case developed for requirements analysis can be utilized in various software development activities such as project organization and testing. Each time the artefact crosses a boundary, its purpose may change depending on the specific needs of the individuals involved. BAs can provide shared meaning by addressing the needs of different social contexts. To accomplish this, they are adaptable enough to meet local requirements while maintaining a consistent identity as they cross boundaries. When widely employed, they possess a loose structure, but they acquire greater structure when used individually to fulfill local needs. These artefacts can take on abstract or concrete forms [175].

In the realm of software development, BAs prove valuable as they can serve stakeholders across different boundaries, particularly in geographically distributed environments. However, if not properly managed, these artefacts can face challenges ranging from being overloaded with multiple interpretations, making it difficult to locate relevant information, to lacking sufficient details altogether. In such situations, boundary artefacts can lose their value to stakeholders and diminish their effectiveness [2, 202, 206].

The creation and management of BAs play a crucial role in shaping people's trust in them. The circumstances surrounding BAs, such as their development and maintenance processes, can influence the level of trust stakeholders have in these artefacts. If the creation and management of BAs are perceived as unreliable or flawed, stakeholders may be hesitant to rely on them, leading to their underutilisation. Moreover, the perceived value and attractiveness of BAs are influenced by their ability to meet the specific needs and requirements of stakeholders.

1.1.4 Discussion on the open gaps

Developing expertise in knowledge resource management can enhance team communication and adaptability to change. The strategies chosen to be implemented can greatly assist in attaining and maintaining a competitive advantage, offering numerous benefits to software development [68, 84, 161, 192]. However, the management of such resources is not deeply understood and investigated [18]. According to Rus and Lindvall [161], the primary cause of failures in KM implementations is the failure of many companies to define clear goals in their strategies. Instead of focusing on managing knowledge, these companies often prioritise the management of documents.

Although software engineering research has progressed in addressing knowledge processes such as storage and sharing, it remains relatively distant from the mainstream research in KM regarding creation and application processes [18].

Existing comprehensive studies strongly focus on knowledge sharing [31,68,205]. Others focus on topics investigated or the main findings on KM in software engineering [18, 30]. Similar to knowledge processes and strategies, studies on identifying knowledge resources are also scarce, and they concentrate mainly on the strategic management field. Evidence from these studies has shown that knowledge resources enhance product and organisational innovation [97, 134, 138].

While it is evident that knowledge resources play a crucial role in securing a competitive advantage from a broad standpoint, it is important to note that each industry domain possesses its own unique set of KBRs. As a result, companies are faced with the challenge of enhancing their competitiveness through diverse approaches tailored to their specific industry context. In 1996, Miller and Shamsie [134] proposed that research should prioritise investigating the influence of KBRs in "turbulent environments," which includes the software industry. However, to our knowledge, such research has not been performed. This thesis contributes to filling the gap, providing more evidence and insights on materialising knowledge resources in software project environments (gap 1).

The effective approach to managing knowledge resources seems to relate strongly to management encouragement, and implementing it involves conducting an internal analysis of the company's resources and identifying its specific needs. Subsequently, the decision-making process involves selecting appropriate tools and methods that align with the defined goals [23]. However, studies discussing formal and comprehensive approaches to managing knowledge resources are rare. The software engineering literature offers little guidance on effectively managing knowledge resources in software project environments (gap 2). As our thesis narrowed down to focus on PBR due to inefficiencies in their creation and management discovered during our investigations, we felt the need to deepen this analysis. We found recommendations and practices to manage PBRs, specifically BAs, our chosen PBR [91, 189, 202]. However, these findings are particular to specific cases.

We incorporate formal and structured aspects of KM approaches to provide formal planning to guide the creation of BAs. We explore the contrast between adopting agile practices (although not explicitly stated in agile principles) and traditional plan-driven organisations. It seems logical to seek a middle ground that incorporates the strengths of both approaches. Introducing more formal planning in managing PBRs does not imply reverting to a documentation-centric approach but instead focuses on effectively creating and maintaining reliable, predictable, and functional artefacts.

1.2 Research Methodology

This section describes the research methodology employed to conduct the studies that comprise this thesis.

1.2.1 Research Questions

This thesis aims to contribute to a better understanding of knowledge resources in agile software project environments and provide guidance on effectively managing them. To address this objective, we formulated the following research questions:

- **RQ1**: *What is the state of reported evidence regarding knowledge resource management in agile software development environments?*

  This first research question contributes to this thesis's theoretical foundation by synthesizing empirical studies to better understand how software companies implement knowledge management strategies and which knowledge processes these strategies support (Chapter 2).

- **RQ2**: *How are knowledge resources managed in agile contexts?*

  The second research question empirically examines knowledge resources in practice. First, we identify the KBRs and how they support software companies with the constant changes in agile environments, revealing inefficiencies when producing PBRs (Chapter 3). Further, we deepen these inefficiencies and explore how the differences in trust levels affect the utilisation of PBRs, specifically boundary artefacts (Chapter 4).

- **RQ3**: *How to improve knowledge resources management in agile contexts?*

  The third research question explores ways to manage knowledge resources. First, we discuss the need for strategic planning for knowledge resources that target KBRs and PBRs, avoiding isolated practices (Chapter 5). We then gather principles that support the creation of PBRs (boundary artefacts) in software engineering and test the solution in an industry context (Chapter 6). At last, we reflect on the knowledge gained through the research performed in this doctoral endeavour and summarise our contributions regarding managing knowledge resources (Chapter 7).

In Figure 1.1, we summarised the alignment between the outlined research gaps with our research questions, also providing the method utilised in the investigation, which we further detail in Section 1.2.

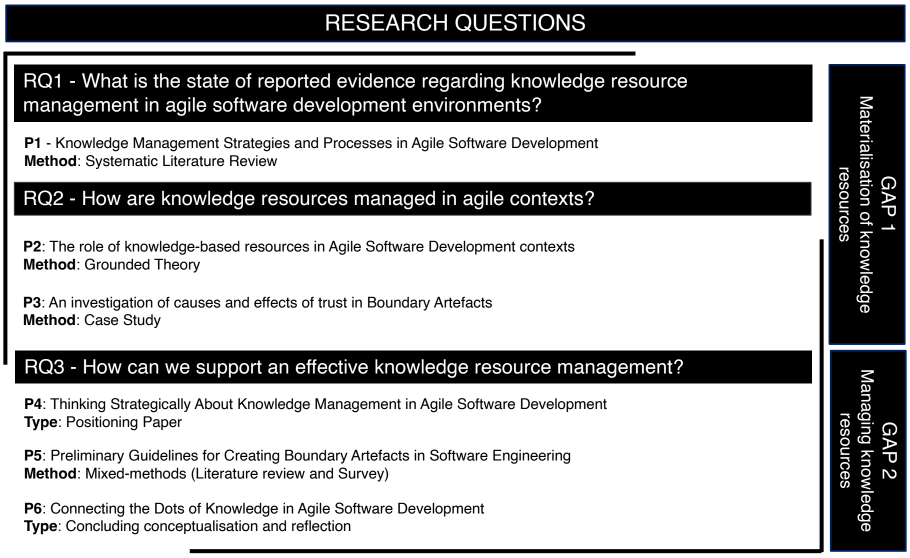

RESEARCH QUESTIONS
RQ1 - What is the state of reported evidence regarding knowledge resource management in agile software development environments?
P1 - Knowledge Management Strategies and Processes in Agile Software Development
Method: Systematic Literature Review
RQ2 - How are knowledge resources managed in agile contexts?
P2: The role of knowledge-based resources in Agile Software Development contexts
Method: Grounded Theory
P3: An investigation of causes and effects of trust in Boundary Artefacts
Method: Case Study
RQ3 - How can we support an effective knowledge resource management?
P4: Thinking Strategically About Knowledge Management in Agile Software Development
Type: Positioning Paper
P5: Preliminary Guidelines for Creating Boundary Artefacts in Software Engineering
Method: Mixed-methods (Literature review and Survey)
P6: Connecting the Dots of Knowledge in Agile Software Development
Type: Concluding conceptualisation and reflection
GAP 1 Materialisation of knowledge resources
GAP 2 Managing knowledge resources

Figure 1.1: Alignment Between Research Questions and Research Gaps. P's stand for papers that help answer the research questions.

We follow mostly a qualitative paradigm to investigate knowledge resource management [159]. In software development, humans create and share knowledge, which will then be expressed into products. Such a mechanism occurs in all communication formats among practitioners, offering the best opportunity for observation [167].

We adhere to social constructivism research, which notes that the construction of knowledge undergoes constant changes and is affected by human interaction [28]. Knowledge is perceived as socially constructed through interaction among people, making it impossible to have a separate existence [159]. This epistemological position is expressed in agile environments where practitioners engage in knowledge processes throughout software development. As qualitative and quantitative methods of investigation, we utilised literature reviews, grounded theory, survey and a case study.

1.2.2 Company Collaboration

A major part of this thesis was conducted in close collaboration with Axis Communications, a company that provides network solutions in video surveillance, access control, intercom, and audio systems. Axis has around 4,000 employees in over 50 countries, and its headquarters is in Lund, Sweden, where two studies reported in this thesis were conducted (see chapters 5 and 6).

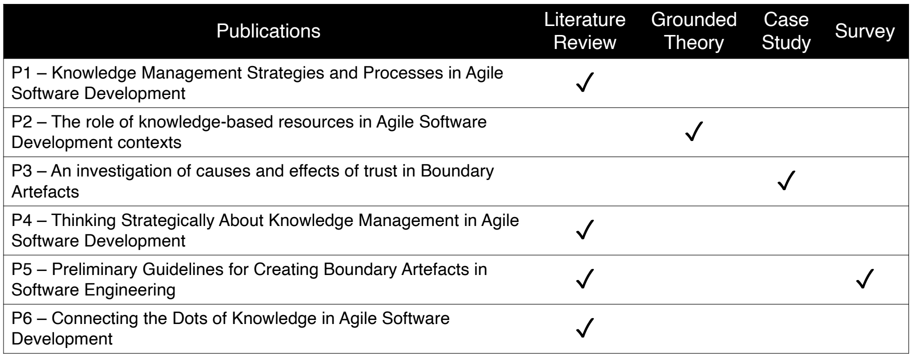
The studies mainly focused on the quality assurance department working with the AXIS OS operating system for edge devices, which is used in more than 200 of Axis' products. Over 1,000 research and development engineers work directly or indirectly with developing AXIS OS as a software product line. There are also other Axis employees and Axis partners that are dependent on information about individual products in the AXIS OS product portfolio, such as technical writers, support personnel and development partners.

The investigation about trust in boundary artefacts (reported in Chapter 4) sparked critical thinking at the company regarding their way of working and their produced artefacts. After revealing how the partial management of BAs had implications for the different stakeholders, the company made changes to their processes and artefacts. They complemented the management of the artefact by covering all parts of the content and designed a new artefact based on their information needs. As the investigated BA could not support more content, risking being overloaded with information, the company has decided to create another BA to supply different stakeholders.

The results of this investigation steered our next study regarding the preliminary guidelines that can support the creation of BAs (reported in Chapter 6). Our curiosity rested mainly on how to increase the trustworthiness of BAs due to their influence on how stakeholders utilise BAs. Besides providing the tools to evaluate our guidelines through workshops and discussions, the company also used them to elaborate their new BA.

1.2.3 Methods

As qualitative methods of investigation, we utilised literature reviews (P1, P4, P5, and P6), grounded theory (P2), and a case study (P3). We also utilised a survey (P5) as quantitative method. Figure 1.2 displays the methods we applied in each publication.



Publications	Literature Review	Grounded Theory	Case Study	Survey
P1 – Knowledge Management Strategies and Processes in Agile Software Development	✓			
P2 – The role of knowledge-based resources in Agile Software Development contexts		✓		
P3 – An investigation of causes and effects of trust in Boundary Artefacts			✓	
P4 – Thinking Strategically About Knowledge Management in Agile Software Development	✓			
P5 – Preliminary Guidelines for Creating Boundary Artefacts in Software Engineering	✓			✓
P6 – Connecting the Dots of Knowledge in Agile Software Development	✓			

Figure 1.2: Research Methods Applied in the Publications.

Literature Review

Reviewing the literature is essential to investigate the existing evidence regarding a researched topic, providing direction and foundations for new research insights [43, 191]. There are different types of literature reviews with their procedures and aimed results: critical literature reviews, state-of-the-art reviews, systematic reviews, scoping reviews, umbrella reviews, realistic reviews, systematic rapid reviews, and qualitative meta-synthesis [159].

We chose to utilise literature reviews in four studies. In our first study (P1), we applied a Systematic Literature Review (SLR), following the guidelines proposed by [100] and complemented by snowballing procedure proposed by [199]. We chose it because we need to summarise the state-of-the-art regarding knowledge management, its application in agile environments, and investigating strategies and how they promote knowledge processes. The results gave us an overview of existing evidence and steered our discussions on future investigations, i.e., identifying KBRs.

We also chose a literature review to gather principles, resulting in our preliminary guidelines reported in P5. We did so to collect published materials regarding practices adopted to create/manage boundary artefacts. Although we created a search string and applied it to a research database, our purpose was not to derive any conclusions from it as in a systematic literature review.

The data for this study (P5) were collected by extracting relevant information from the studies we selected (based on selection criteria). We organised the information in a way that helped us synthesising the evidence and draw conclusions.

Similarly to the two studies mentioned above (P1 and P5), we also utilised literature reviews to base our discussion on our vision paper (P4) and gather evidence of studies performed in the industry to develop our work in P6.

As **data analysis**, we used quantitative analysis, specifically descriptive statistics and visualizations, to explore and summarize the studies. We applied narrative synthesis to answer the research questions [155].

Grounded Theory

Grounded theory is a method that seeks to generate a theory that emerges from the data collected by investigating an unexplored phenomenon [45, 159]. The reasons we chose to utilise this research method are twofold. First, studies on Knowledge-based resources are scarce and inexistent in the software engineering literature [134, 147]. Second, the grounded theory methodology supports the generation of theoretical explanations and offers new insights regarding a particular phenomenon. This method allowed us to identify the knowledge-based resources and understand the phenomena that lie behind the use of these resources in software development. In our study, we developed the Knowledge-push theory, which reveals how KBRs boost the Need for Change in ASD. We followed the principles for theory generation established by Corbin and Strauss [45].

The data for this study were collected by means of interviews. We asked the questions according to the role of each practitioner who was interviewed, be it a product owner, scrum master, line manager, or project manager. A few practitioners presented archive documents during the interviews, such as internal presentations displaying the distributions of the teams, the quality framework adopted, and organisational schemas. However, the documents served as contextualisation, facilitating our understanding of some of the topics discussed during the interviews.

The **data analysis** followed the steps included in the grounded theory method for generating theory from the data through the following coding process: open coding, axial coding, and selective coding [44]. We also observed the guidelines proposed by Stol et al. [179] for reporting our study.

Case Study

As we discovered the inefficiencies in utilising knowledge resources through our grounded theory study, we decided to go further and explore, more specifically, why practitioners did not trust the artefacts they produced. We chose the case study method for two main reasons. First, we wanted to narrow down our investigation to artefacts with a deep analysis. Second, our industry partner was also interested in investigating issues with one or their artefact, which created a favourable environment for applying the case study method.

Case studies are the type of research method most reported in knowledge management studies in software engineering [148]. It refers to a method that aims to "investigate a contemporary situation in their context, utilising multiple sources of evidence" [160]. They do not often generate the same results (generalisability), which is not their purpose, but they offer great potential for a deeper understanding of a particular situation. We focused on an exploratory case study to gather insights into creating and utilising the selected BA [160].

The data for this study were collected by means of interviews, archive documents, workshops, and informal conversations with practitioners. We utilised semi-structured interviews and one unstructured (due to the exploratory character of the event).

The **data analysis** was interrelated with the data collection. We performed a two-cycle coding [133]. In the first cycle, we assigned process codes to the data chunks. This coding method indicates observable and conceptual action in the data and summarises data segments. In the second cycle, we broke down the codes generated in the first cycle into smaller concepts that represented patterns in the data. After concluding the second cycle, the first and the second authors gathered to analyse the generated codes. To increase reliability, we conducted a workshop with practitioners participating in the study at the company partner Axis Communications. We asked for confirmation of our findings in relation to the three aspects of our approach, including content creation and management of the BA, the implication of trust beliefs, and practitioners' behaviour changes. After the workshop, we adjusted our interpretations.

Survey

Survey is a research method involving several steps to gather information about a particular population or problem under investigation [96]. We applied this method as part of our investigation to elaborate the guidelines for creating boundary artefacts in Software Engineering (P5). We chose this method because we wanted to gather the condensed perception of practitioners who participated in the study workshops.

We followed Kasunic [96] seven-step process to design and report our survey.:

- *Research objectives*: Condense the perception of the practitioners that participated in the workshops regarding how the guidelines' principles contributed to the trust factors' reliability, predictability and functionality. Also, to evaluate the guidelines' usefulness, ease of use, and intent to use.
- *Identify and characterise target audience*: The survey had as a target audience the participants in workshops we performed to validate our guidelines.
- *Design sampling plan*: Our population was the six people participating in the study.
- *Design and write questionnaire*: We designed our questionnaire based on the trust factors established by Lansing and Sunyaev [108] and theoretical model for validating information system design methods [135] to evaluate the ease of use, usefulness, and intention to use.
- *Pilot test questionnaire*: We tested our questionnaire and adjusted it to understand the questions better. The adjustments focused more on rephrasing the questions, but the trust factors were discussed during the workshop before sending out the questionnaire.
- *Distribute the questionnaire*: We sent the questionnaire to all participants via institutional e-mail. However, their responses were anonymous.
- *Analyse results and write report*: The data analysis from the survey was done through descriptive statistics, focusing on the frequency of the responses. We also utilised stacked bard charts for visualisation.

1.2.4 Threats to validity

In this section, we discuss the threats to the validity of this thesis, while each publication has its detailed discussion on the matter. We discuss threats regarding generalizability, validity, and reliability for qualitative research provided by Leung [117].

Validity. This threat refers to the appropriateness criteria of the tools, processes, and data. The choice of the methods must enable the detection of the findings to be valid in a certain context. For the methods we minimised this threat by applying methods that helped us gather insights about unknown phenomena, such as knowledge-based resources through grounded theory. Also, to deepen our understanding of how people behaved when differences in trust happened by applying exploratory case studies. For extracting data, we utilised several processes that increased the validity of our findings. For example, discussion for validating thoughts and insights among the study's authors, workshops with company partners to ensure our interpretations were correct, and tools to store and process qualitative data (audios of interviews).

Reliability. For the quantitative methods, reliability refers to how replicable the studies are. We applied a low number of quantitative methods compared to the qualitative ones. To minimise the threat of not having replicable studies, we provided entire protocols for further replications (reported in Chapter 2) and questionnaires (reported in Chapter 6).

In qualitative methods, the threat lies in consistency. According to Leung [117], qualitative research is expected to have a margin of variability. However, there are techniques to reduce such events. For better accuracy, we applied constant comparison (reported in Chapter 3 within the data and among co-authors. To ensure consistency, we also performed a 1st level triangulation, which is a cross-check among authors, data and company (reported in Chapter 4 and Chapter 6).

Generalisability. This threat refers to the analytical generalisation in qualitative studies, where the findings of one study can be generalised to another within the same social context, place, people, and time. In our thesis, we aimed for generalisability in two studies (reported in Chapter 2 and Chapter 3). In Chapter 2, we tried to increase generalisability: by creating broad search strings and complementing database search with the snowballing procedure to capture a high number of peer-reviewed studies in databases not covered by the ones we utilised. In Chapter 3, we diversified our sampling with practitioners in different roles, countries, and organisation domains, aiming for a broad perspective concerning the contexts.

For the case study (Chapter 4) and the study involving a mixed-method approach (Chapter 6), we cannot claim the transferability of the findings to other domains. In the specific case of these studies, we claim analytical generalisation through detailed context information and discussion of the findings.

1.3 Summary of Results

This section summarises the results of our six publications. We explain how the results address our research questions (see Subsection 1.2.1). Figure 1.3 displays the sequence, discoveries and insights that led to the execution of the studies.

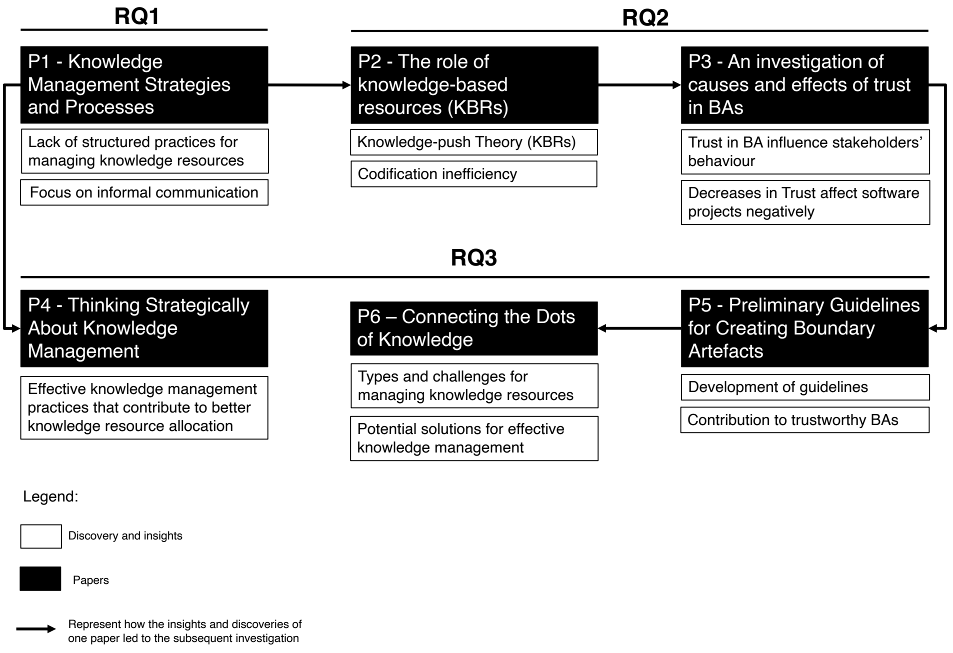

RQ1
RQ2
RQ3
P1 - Knowledge Management Strategies and Processes
Lack of structured practices for managing knowledge resources
Focus on informal communication
P2 - The role of knowledge-based resources (KBRs)
Knowledge-push Theory (KBRs)
Codification inefficiency
P3 - An investigation of causes and effects of trust in BAs
Trust in BA influence stakeholders' behaviour
Decreases in Trust affect software projects negatively
P4 - Thinking Strategically About Knowledge Management
Effective knowledge management practices that contribute to better knowledge resource allocation
P6 – Connecting the Dots of Knowledge
Types and challenges for managing knowledge resources
Potential solutions for effective knowledge management
P5 - Preliminary Guidelines for Creating Boundary Artefacts
Development of guidelines
Contribution to trustworthy BAs
Legend:
Discovery and insights
Papers
Represent how the insights and discoveries of one paper led to the subsequent investigation

Figure 1.3: The sequence of execution of studies.

RQ1: *What is the state of reported evidence regarding knowledge resource management in agile software development environments?*

Knowledge Management Strategies and Processes in Agile Software Development: A Systematic Literature Review (P1). Through this review, we identified the proportions of the utilisation of knowledge management practices reported in empirical studies in agile software development. We identified a ratio of 81% of practices following personalisation and 19% following codification strategies. Although we found many practices, most focusing on knowledge sharing, we could not find an overarching knowledge management program targeting knowledge in different formats. For example, establishing communities of practices with the ultimate goal of brainstorming with people from different departments is part of our knowledge management program that focuses on developing a new product in our product line. Some practices seem to be isolated and more of a side effect than an intended purpose, i.e., pair programming and customer interaction. Moreover, the intense focus on informal communication can result in the loss of a significant amount of knowledge which is not properly transferred to other individuals. Instead of propagating the knowledge, it remains inside a few individuals' minds. These results addressed our RQ1 by providing a big picture of knowledge management strategies and processes that these strategies support. But more importantly, it gave us an overview of the remaining issues that led us to the next step: the need to understand how knowledge resources translate in agile environments.

RQ2: *How are knowledge resources managed in agile contexts?*

The Role of Knowledge-based Resources in Agile Software Development Contexts (P2). The results of our systematic literature review steered this next study. We conducted a grounded theory study (See 1.2.3) in the agile software development context. As part of our investigation, we needed to understand better the types of knowledge resources utilised in agile environments. We identified many KBRs in their different forms, including specific skills. These KBRs were then gathered together into the Knowledge-Push Theory. This theory provides an explanation of how practitioners use the identified KBRs to boost the need for change in ASD. The results of our study show that, as primary strategies, practitioners use KBRs for task planning, resource management, and social collaboration. These strategies are implemented through the team environment and settings and are made manifest in the practitioners' ability to codify and transmit knowledge. However, this process is non-systematic, which brings inefficiency into the domain of knowledge resource utilisation, resulting in potential knowledge waste. This process can generate negative implications for the course of software development, including meaningless searches in databases, frustration because of recurrent problems, the unnecessary redesign of solutions, and a lack of awareness of knowledge sources. These results have caught our attention to the creation and utilisation of artefacts, specifically why people do not trust the artefacts they create.

An investigation of causes and effects of trust in Boundary Artefacts (P3). To deepen our understanding of trust in artefacts, we conducted an empirical investigation through a case study to reveal how people create, utilise and manage boundary artefacts. We observed how differences in trust affected software projects and how they influenced the stakeholders' behaviour. Our investigation has shown that practitioners who add new and adjust existing content do not entirely understand the stakeholders' needs. In addition, the management of the content is partial. These findings indicate that parts of the content and stakeholders' needs are not supervised, impacting trust levels. When practitioners do not perceive the boundary artefact as favourable in at least one of the investigated trusting beliefs, specifically reliability and predictability, the stakeholders cannot execute their tasks appropriately, and several implications affect the software project development. Additionally, the stakeholders create workarounds to supply their needs. Among those, they create complementary artefacts within their teams. Organisations need to increase the level of trust in their boundary artefacts to function as planned, and the proper management of the content is crucial for increasing this trust level. This management should include creating content based on stakeholders' needs and monitoring changes.

RQ3: *How to improve knowledge resources management in agile contexts?*

Thinking Strategically About Knowledge Management in Agile Software Development (P4). Since knowledge plays an essential role in software development, it is important to have effective knowledge management practices that contribute to better knowledge resource allocation. In this vision paper, we discuss potential approaches to managing knowledge resources. We base ourselves on strategic management, emphasising that the means by which a company allocates and uses its resources confers the competitive advantage that differentiates it from its competitors. Knowledge is a resource that requires comprehensive and logical management to obtain benefits since, together with skills, it is the main resource of software development organisations. This discussion inspired us to move towards utilising the knowledge-based resource theory for raising the importance of this resource and providing means of utilising it strategically.

Preliminary Guidelines for Creating Boundary Artefacts in Software Engineering (P5). To contribute to a strategic approach to managing knowledge resources (specifically PBRs), we developed preliminary guidelines that advise creating BAs. in addition, through an expert validation of the guidelines, we collected practitioners' perceptions on how each guideline's principles contribute to creating trustworthy BAs. We grouped the principles collected from a literature review into three categories. The first category (Scope) focuses on the scope, displaying principles referring to defining each boundary's target audience, needs, and terminology. The second category (Structure) relates to how the artefact's content is structured to meet stakeholders' needs. The third (Management) refers to principles that can guide the establishment of practices to manage the artefact throughout time. The expert validation revealed that the principles contribute to creating trustworthy BAs at different levels—also, the relevance of the guidelines and their usefulness. Moreover, the guidelines strengthen BA traits such as shared understanding, plasticity and ability to transfer. Practitioners can utilise the guidelines to guide the creation or even evaluate current practices for existing BAs.

Connecting the Dots of Knowledge in Agile Software Development (P6). We developed this conceptual paper to gather in a concise format the evidence that we collected through this doctoral journey and offered a simplified discussion about knowledge resources in an agile context. We explore their types, challenges and potential solutions to effectively manage knowledge, especially what is stored in artefacts.

1.4 Discussion

This section discusses our findings and describes their contributions to the identified gaps (see section 1).

1.4.1 Materialisation of knowledge resources in software project environments

This dissertation is set in the cross-section of organisational studies and software engineering. Our contribution is incorporating organisational principles into the software engineering literature in agile contexts.

First, we summarise and analyse the state-of-the-art literature about knowledge management, its application and its relation to agile software development. We illustrate how knowledge management strategies are distributed in the hierarchical layers of software development companies that adopt agile methods through a conceptual classification framework. The insights derived from this systematic literature review provide comprehension concerning how the companies implement knowledge management strategies with practices that promote the knowledge processes [148].

Second, we base ourselves on the knowledge-based resource theory to identify the KBRs and how they are distributed in agile software development projects. We contextualise these KBRs in the Knowledge-push Theory, illustrating how they boost the need for change in ASD environments. Each theory category of the theory highlights different contributions to existing software engineering aspects. For example, software product management receives a contribution from the category Scenario Analysis. The behavioural software engineering area receives contributions from Social collaboration and Team environments and settings categories. The resource management aspect receives contributions from Task planning and resource management. The coordination aspect benefits from the category Ability to systematise and transmit knowledge [147].

Third, we uncover the process of creating, utilising, and managing boundary artefacts in software projects. We establish a conceptual idea of how the trust cycle of boundary artefacts functions. We state that trusting beliefs about the boundary artefact influence how the stakeholders use the boundary artefact. Negative implications might emerge when practitioners do not perceive the BA as favourable in at least one of the investigated trusting beliefs, reliability, functionality, and predictability.

Thus, the trust cycle is generalisable to different software development companies with different sizes. The negative implications resulting from the inconsistencies, though, can be the same or others because they are conditional to the context. For example, we did not find implications regarding the functional factor. Other contexts or different BAs can reveal them — also, different behaviours [145].

1.4.2 Guidance on how to effectively manage knowledge resources in software project environments

This thesis also contributes to providing guidance on managing knowledge resources. We specifically introduce a resource management perspective for managing KBRs and PBRs. We argue that as a knowledge-intense activity, software development is significantly dependent on exploiting knowledge resources. However, that implies that it must be managed logically in a structured manner [11, 18, 161].

Our focus on transforming KBRs into PBRs requires further development, as we revealed inefficiencies that can have negative implications for software projects. Managing knowledge through PBRs requires more planning and structure than normally associated with agile software development contexts. However, we need to think about the changes software companies have endured since the wide adoption of agile principles and all the principles and practices introduced since then. Companies have grown and spread development activities across countries. Artefacts have become a key to enabling communication and coordination, as well as being a critical resource for executing tasks.

Our work expands on previous research on boundary artefacts by providing a structured guideline gathered into three main categories (scope, structure, and management) that help practitioners create BAs in software engineering contexts. This guideline is significant and unique because it offers an original perspective on essential principles that practitioners should observe upon conceiving a boundary artefact. We incorporated extendable practices applied to specific contexts, such as product family development and the automotive industry [91, 202]. These were considered relevant for the context we utilised in this study, which we believe has a potential for generalisation of the principles as more applications of the guideline may occur.

Lastly, we discuss the dichotomy between agile adoption (albeit not formally expressed in agile principles) and traditional plan-driven organisations. Combining the best of both views seems reasonable. Dybå and Dingsøyr [60] suggest exploiting traditional planning principles in some cases, especially for greater efforts. Similarly, including more formal planning in managing PBRs does not mean moving back to a documentation-centred approach, but rather managing the creation and maintenance of reliable, predictable and functional artefacts.

1.5 Contributions, Implications and Future Research

This section reports the contribution of our thesis to the software engineering literature, the implication of our findings for practices, and which future questions we leave open.

1.5.1 Contributions to Research

This thesis, in general, contributes to an increased understanding of the translation of knowledge utilisation in agile software development environments. Our findings have shed light on the different knowledge processes (creation, sharing, storing and application) researchers have focused on when studying agile contexts (P1). Moreover, the findings have revealed, *first*, the connection between knowledge strategies and processes (P1), which made it possible to see that the practices will benefit one or more processes regardless of the strategy adopted. For software engineering and other literature areas, such understanding is important as it gives a better overview of how those concepts connect, what was not apparent before.

Knowledge is well recognised as critical for software development, especially for coordinating dependencies [18, 129, 161]. It is true, though, that it is not easy to research due to its vague trait, which explains the vast number of descriptive studies reporting practices targeting knowledge sharing (SLR), while more deep and longitudinal studies focusing on explaining, for example, knowledge creation and application are scarce. The *second* contribution of this thesis targets this complexity by incorporating into software engineering literature the concepts that originated in the resource-based view of the firm [11], focusing on knowledge resources (P2). Optimisation and efficient allocation of resources are concepts that historically relate to the manufacturing industry. But this thinking is also valid in developing software-intensive products and services. However, the proportions of the type of resources have changed. Such understanding is relevant for the software engineering literature as it reveals how tangible and intangible knowledge proportions remain. We detail this through the Knowledge-based resource work from Millier and Shamsie [134].

The proportion of employees' knowledge transferred to artefacts (PBRs) is smaller than what remains intangible in agile software development contexts. However, in our grounded theory study (P2), this transfer process has been proven to be quite inefficient, making practitioners sceptical about utilising some produced artefacts. We wanted to better understand why they could not trust the artefacts produced. However, the literature is scarce and did not focus specifically on the type of artefacts we wanted to explore. In this regard, our *third* contribution to software engineering literature was to expand on the work from Lansing and Sunyaev [108] on trust in inanimate software artefacts by examining the differences in trust in boundary artefacts and how they can affect software projects and stakeholders' behaviour (P3). This unique study has provided new insights into the trust cycle in inanimate software artefacts. As the boundary artefact carries many inconsistencies, the stakeholders can show less confidence, reducing their trust in it. They will use it differently from how it was supposed to be used or not use it all.

Fourth and lastly, our research results contributed to the software engineering field, synthesising the literature on the principles that can guide the creation of boundary artefacts (P5). We explored reported guides and practices in different subjects and validated them in industry settings, revealing practitioners' perceptions on how the guidelines contributed to creating trustworthy boundary artefacts.

In summary, the findings of this thesis contribute primarily to research on knowledge management in agile contexts, knowledge resources, and trust in inanimate software artefacts. While focusing on agile contexts, the thesis also contributes to the intersection of different research areas, such as organisational studies, connecting and applying well-established concepts and theories (knowledge and resource management) to the software engineering field.

1.5.2 Implications for Practice

The findings of our thesis have several implications for practitioners in software engineering. We start by mentioning that practitioners can utilise the results to understand how the practices adopted to manage knowledge trigger at least one of the existing knowledge processes (creation, transfer/share, storage, and application). We demonstrated the importance of aligning knowledge management practices to a company's overall strategy for the purpose of efficacy measures and follow-up on general contribution to economic value production. It is hard to grasp their efficacy when practices are reported as isolated or do not connect to the rest of the company. Similarly, without a pre-established goal, it is complex to see the outcomes.

Second, our findings help practitioners identify the types of knowledge resources displaying them in a concrete perspective and helping to understand such an abstract concept as knowledge. We explain how knowledge-based resources support agile companies in embracing and implementing changes. Additionally, we identify these resources and how they manifest into technical, creative, collaborative, and integrative skills. Additionally, we provide insights into how inefficiencies in storing knowledge can generate waste.

Third, practitioners can see and understand the implications of decreased trust and how they influence the stakeholders' behaviour towards boundary artefacts. As the stakeholders rely on boundary artefacts to execute their tasks, they expect the information stored in the artefact to fulfil their needs and meet trust factors such as reliability, predictability and functionality. When the content stored in these artefacts does not meet these factors, we found that several implications can affect the development process, which can help practitioners understand why people have certain behaviours towards boundary artefacts.

Fourth, we offer practitioners guidelines focusing on the creation of boundary artefacts. The lack of management and a structured process for creating such artefacts reduce the confidence of stakeholders in them. To help with that, we show how to generate boundary artefacts considering principles condensed into three main categories: *Scope* of the artefact, *Structure*, and *Management* practices.

1.5.3 Future Research

This section provides a few directions for future research. While the studies cover specific research directions from what is investigated, we now elaborate on general directions to be considered in the intersection between knowledge management and software engineering.

Investigating mechanisms and practices for measuring the efficacy of knowledge resource management practices. Our thesis has shown how companies utilise practices that follow personalisation or codification strategies for managing knowledge resources in ASD. However, further research is needed to understand how these practices are evaluated and if they are effective, which triggers the need for more longitudinal studies. Moreover, facilitating means for evaluating practices that target KBRs or PBRs is welcome as they help pragmatically introduce such concepts in software engineering.

Studying and developing lightweight knowledge management programs for software projects. Further research could expand our contribution by investigating how to implement knowledge management programs (targeting personalisation and codification strategies) in software projects that connect to the overall companies' resource management strategies. We could not find such studies, and formalising such programs could contribute to contextualising knowledge management practices that seem to be isolated from the rest of the software companies [148].

Studying waste production in the transformation of KBRs into PBRs in software development. While we provided evidence of a series of inefficiencies in transforming KBRs into PBRs, future research could dive into this aspect to explore how much waste is generated due to an inefficient process. It is known that PBRs can be an economic value source as they become the companies' property and are utilised to generate profit. Therefore, such investigations could shed light on how much knowledge is being lost and how much economic value is being wasted when the proper transfer of KBRs to PBRs does not occur.

1.6 Conclusion

Agile principles have rapidly spread among software companies that needed to incorporate product changes in several phases of their software projects. Among several principles, agile greatly encourages informal communication and reduces focus on documentation. This phenomenon uncovered the importance of knowledge as a critical resource in assimilating these changes. However, managing such resources comes with challenges, as knowledge is predominantly tacit. In this thesis, we concentrate on knowledge resources, their identification, operation and, more deeply, how they manifest in boundary artefacts.

Knowledge management focuses on managing an organisation's workforce through social processes that facilitate interaction among people. We start our investigation by examining the existing evidence from the literature to show the proportions of knowledge management practices utilised in ASD environments and the knowledge process they focus on (RQ1). After examining the results, we observed that practices and processes were formally isolated from the rest of the companies and lacked effective evaluation. We decided it was necessary to investigate further the existing KBRs in ASD and potential organisational theories that could guide our contributions towards the end of this PhD.

As we progressed in our investigation (RQ2), we identified KBRs that support changes in agile environments in the Knowledge-push theory through a grounded theory study. This theory explains how practitioners use the identified KBRs to boost the need for change in ASD. KBRs often manifest as specific skills, including technical, creative, integrative and collaborative skills. In this same study, we collected surprising results regarding inefficiencies in converting KBRs into PBRs. These results greatly impacted the direction and how we narrowed down our research. We observed that even though the practices for codification strategies were reported in minor numbers (reported in our first study, the SLR), the inefficiencies in these practices had a major impact on the practitioners' task execution and were reported by many practitioners.

Regarding proposed solutions (RQ3), we condensed our potential solutions into two main studies. First, we condensed evidence from empirical papers carried out in the industry to match six main challenges related to the creation and management of PBRs. Second, we developed a preliminary guideline that supports the creation of BAs in software engineering, which was perceived as being able to increase the trustworthiness of BAs.

The empirical evidence gathered in this thesis contributes to the software engineering literature by incorporating knowledge resource management literature into BAs in agile software development projects. Each chapter provides important findings for both researchers and practitioners. Adopting the resource perspective to knowledge clarified the limited possibilities of codifying knowledge, explaining that a large proportion of knowledge remains tacit and can be seen through the KBRs, which we revealed in our Knowledge-push theory. The codifiable portions of knowledge are usually stored in artefacts in various formats and shared among practitioners. As we focused on BAs, our findings facilitated the understanding of the necessity of common understanding between creators and stakeholders. The provided guideline and trust cycle clarification offer relevant assistance to practitioners utilising BAs as PBRs. Lastly, including more formal planning in managing PBRs does not mean moving back to a documentation-centred approach but rather managing the creation and maintenance of reliable, predictable and functional artefacts.

Chapter 2

Knowledge management strategies and processes in agile software development: A systematic literature review

This chapter is based on the following paper:

> Raquel Ouriques, Krzysztof Wnuk, Tony Gorschek, and Richard Berntsson Svensson. "Knowledge Management Strategies and Processes in Agile Software Development: A Systematic Literature Review" *International Journal of Software Engineering and Knowledge Engineering*, 29(3): 345–380, (2019)

2.1 Introduction

Software-intensive product development companies struggle to stay competitive due to fierce competition and increased pressure, forcing them to quickly release new products to the market [136, 157]. These forces push companies to improve resource management aiming for better product quality, creativity or efficient development process [50]. Many companies have also introduced Agile Software Development (ASD) methods, as they are perceived to offer better response to frequently changing market needs, more flexible software development methods and shorter learning cycles [42].

The increased flexibility that ASD promises comes with the cost of prioritizing informal communication between team members over written documentation [39] or loss of the “big picture” of the product due to extensive focus on developing features [8]. As a result, managing the knowledge asset becomes critical, and the lack of it can lead to several negative effects. For example, barriers to collaboration and asynchronous communication in large companies that introduced ASD [7], and competitiveness loss [134].

Mastering how to manage knowledge as a competitive asset [3] can improve team communication and the change responsiveness. The utilization of Knowledge Management (KM) strategies can significantly support achieving and sustaining competitive advantage [84, 192] and brings several benefits to software development [69, 161], e.g. that effective networks for tacit knowledge sharing in ASD contribute to continuous process improvement [19].

How to manage knowledge in ASD is still not well understood or investigated [19]. Rus and Lindvall [161] affirm that the reason for failures in KM deployments happens because many companies do not establish their goals in the KM strategy, and manage documents instead of knowledge.

KM focuses on the effort to manage an organizations’ workforce through social processes that facilitate interaction between individuals [84]. KM also has established principles for dynamic activities that focus on the Knowledge Processes (KPs): creation, storage/retrieval, transfer, and application [3].

Knowledge processes often use practices that implement codification and/or personalization strategies; which should be rooted in the company’s corporate strategy [24], for example, physical open spaces, which implement a personalization strategy, could promote knowledge transfer between individuals in physical structure that facilitates the working together [164]. Although the research literature in ASD encompass several studies on practices for specific KPs [23, 164]; what is lacking, however, is understanding how KM strategies relate to the different KPs in companies adopting ASD.

While the software engineering field had addressed the storage and retrieval knowledge process, the field is still distant from the KM mainstream research, and processes such as knowledge creation, transfer and application remain largely unexplored [18]. This study contributes to filling this gap by means of focusing on how companies adopting ASD manage the knowledge asset implementing KM strategies, focusing on the KPs that they promote in the different organizational layers.

The contribution of the study is threefold: 1) Summarize and analyze the state-of-the-art of the literature in relation to KM, its application and relation to ASD, 2) Illustrate the KM strategies in the hierarchical layers of software development companies that adopt ASD, through a conceptual classification framework, and 3) Provide comprehension concerning how the companies implement KM strategies with practices that promote the KPs.

The rest of this paper is organized as follows: Section 2.2 introduces the background on ASD, KP, KM strategies and the conceptual classification framework. Section 2.3 describes the procedure for conducting this systematic literature review (SLR). Sections 2.4 2.5, and 2.6 report the results, analysis, and discussions. In Section 2.7, we discuss threats to validity, and in Section 2.8 we provide conclusions and implications.

2.2 Background

In this section, we shortly present basic concepts of KM, explaining the types of knowledge, the knowledge processes, and the KM strategies. We provide a brief overview of ASD and a summary of the previous literature reviews in the area. We also compare them to our study, explaining the major differences.

2.2.1 Knowledge management

The challenge of allocating the knowledge asset logically is related to the dilemma of extracting and using knowledge that are distributed in many individuals' minds [80]. Although knowledge has its own theoretical definition, it is commonly used interchangeably with information. According to Nonaka [139], information is a collection of a particular arrangement represented by a flow of messages, whereas knowledge is the meaning created by the combination of information and individuals beliefs.

Knowledge is context dependent, dynamic and created through social interactions between individuals. Knowledge is divided in tacit – rooted in individual's mind and it is a result of values, beliefs, life experiences, emotions, procedures, actions and routines; and explicit – it is easy to transmit since it is already systematized in data, formula, manual, books, specifications, along with others [79,141].

Even though knowledge is considered as the primary source of sustainable advantage for several companies [50], managing it remains challenging. Different fields have studied knowledge Management (KM) due to its relevance and interdisciplinary nature and proposed many concepts, theories, and applications. The term started to appear in academic publications in 1986 [198].

In this study, we adopt the definition of knowledge management (KM) as the effort to manage organizations' workforce through information and communication technologies or creation of a corporate culture that focuses on social processes that facilitates the sharing between individuals, aiming to reach a sustainable source of advantage [84, 192].

KM strategy is the effort to formulate plans for actively manage knowledge [182]. The strategies can be codification and/or personalization strategies, which are rooted on the company's corporate strategy [23]. Codification involves the storage of knowledge into databases for further use (e.g., wiki-based tools [92]), while personalization focus on human interaction to communicate knowledge (e.g., communities of practice) [78]. The KM strategies promote the set of knowledge processes that represent individual's cognitive, social practices and the culture that shapes knowledge in organizations [3]. They are four KPs:

- *Knowledge Creation* (KC) comprises the development of new ideas, concepts or knowledge replacement by the constant combination between tacit and explicit. A critical step to trigger this process is social interaction, which enables individuals to share and develop new knowledge [139]. In ASD, we could exemplify this process being triggered by pair programming, where developers elaborate create new knowledge to complex problems solution based on interaction;

- *Knowledge Storage and retrieval* (KS) is related to the organizational memory, how it keeps the knowledge through documentation, databases, networks of individuals and so on. This memory is built with past experiences, events, and procedures that affect the organization's current activities. For example, inside agile teams, it is common utilize wiki-based tools to store additional documentation related to meetings and problems solution developed by the team. In SCRUM framework, retrospective meetings could support KS;

- *Knowledge Transfer/sharing* (KT) refers to transfer knowledge to areas needed within the organization. The knowledge disperses in different levels, between groups, individuals, across groups, and from the group to the organization. In this process, the main challenge is to know what knowledge the organization need to share, because usually, they do not know what they know. In ASD context, rotating individuals facilitate the transfer of knowledge between team members [22]. Wiki-based tools is another example of transferring knowledge about the software being developed from agile teams to different departments in a company [58];

- *Knowledge Application* (KA) indicates the use of the knowledge as a competitive advantage through improvements in organization capability. Three mechanisms can integrate the knowledge: directives – the collection of rules and procedures; organizational routines – regards to coordination patterns and tasks development; and self-contained task – created especially for problem-solving, when there is no support from directives or organizational routines. In software development companies that adopt SCRUM framework, review sprint and retrospectives might be a good support for KA activities by effectively applying knowledge gained from previous sprints and projects.

A knowledge asset is a knowledge-based resource, for example, individual or team solutions developed for complex problems, related to feature development, based on individuals' previous knowledge and experience. Managing knowledge assets increases the return and also keeps the continuing advantages of new generated, transferred and applied knowledge [3, 50]. In any organization, knowing the knowledge needs is one of the main steps for an effective KM, and also understand that filling repositories with knowledge is not the best approach to have favorable outcomes [120].

To process and apply knowledge is an intuitive activity, derived from mental mechanisms that are capable of connecting and merging conflicting knowledge [139,141]. Since knowledge is a large part of the resources used to build software products, it is crucial to know the cognitive structures process knowledge.

According to Robillard [158], there is a lack of understanding of how knowledge is processed in cognitive structures. Additionally, this process cannot be entirely automated. Different strategies are adopted aiming to codify the knowledge to automate managing it. On the other hand, several strategies focus on stimulating the cognitive structure of individuals through social interactions (personalization). Depending on the organizational goals, a combination of these strategies may enhance one or more KPs. Considering that software development is a knowledge-intensive activity [10], that combines tacit and explicit knowledge during its lifecycle, it is valuable to understand how these constructs are interrelated to apply a good strategy combination.

2.2.2 Agile software development

Traditional software development processes struggle to offer the required flexibility and change responsiveness [83]. Promptly responding to changes is the prevalent idea of ASD [14] that had its principles disseminated in 2002. ASD principles arise as a contradiction to the traditional software development which is inspired by the project cycle and guided by tasks and documents [137]. Instead of extensive documentation to guide the product development, the ASD relies on the team capability of communicating, collaboration, and being creative [50].

Several ASD methods were suggested based on the principles established in 2001. Dybå and Dingsøyr [60] point out six primary agile methods: Crystal methodologies; Dynamic software development method (DSDM); Feature-driven development, Lean software development, Scrum, and Extreme programming (XP; XP2).

Successfully adopting ASD requires several adaptations, since it highly depends on the context that is applied. Contextual factors such as system size, customer demand for software, team expertise, and highly constrained quality attributes requires evaluation in agile adoptions [104]. A flexible organizational structure is also required for agile software development, where collaboration takes place.

Agile software development methods have been adopted in small collocated teams and also in large companies that have distributed teams [32, 122, 150]. Studies had already reported success on this adoption, although several challenges remain. Those challenges are related to team coordination, requirements definition, communication and knowledge sharing [112, 150].

Traditional software development focuses mainly on explicit knowledge while in the agile context, the knowledge management activities are focused mainly on tacit knowledge [18]. Tacit knowledge is difficult to articulate since it is embodied in peoples' mind; it is a result of values, beliefs, and experiences [139]. In addition to this complexity, there are two notable obstacles for managing knowledge asset in software companies: time to transform the tacit knowledge into explicit and use it, and time pressure faced by project managers to deliver running code. However, benefits originated from KM could lead to a better competitive advantage when the knowledge has a crucial goal [10, 57, 144].

2.2.3 Previous systematic literature reviews

We identified five literature reviews that explore knowledge management and software development.

Bjørnson and Dingsøyr published a review [80] involving knowledge management in software engineering in 2008. They reported empirical studies on knowledge management in software engineering, including knowledge management concepts, major findings, and research methods. They observed that tacit knowledge is the focus of knowledge management activities in ASD, while explicit knowledge is more addressed in traditional development processes. They noticed that there were more lessons learned studies (or industrial case studies) than scientific contributions. Recommendations from the authors suggest that it is necessary to explore how to manage tacit knowledge.

In 2013, Camacho et al. published a systematic literature review [31] focusing on research on knowledge transfer in software engineering. They explore how to imply, how to measure, and how it occurs in ASD. Knowledge transfer is performed mainly through technology, using repositories to store information about the work that done and tools to increase collaboration between distributed sites. According to their findings, the conventional measures relates to the amount of innovation and number of new concepts successfully generated. It can also be measured with different rubrics, such as sales, customer satisfaction, number of customers, the value created by research and development activities, and return investment. They claim that the intensive communication facilitates knowledge transfer in the agile environment due to less documentation generation, the focus is on the interaction between people instead of processes and tools.

In 2014, Cabral et al. conducted a systematic literature review [203] about knowledge management in agile methodologies. They considered empirical and non-empirical studies in their analysis. They identified seven topics in the selected papers: project documentation, transfer and collaboration of tacit knowledge, adoption of knowledge management methodology, tools for knowledge management, human and social factor, use of communities of practice, and knowledge artifacts and experience knowledge. Regarding the software development cycle, the authors suggest that adopting knowledge management practices not only change the development cycle but also increase its costs. Most of the tools for knowledge management focus on communication between participants, which, for them, implies on the need for more studies where tools are more focused on the software development process.

In 2015, Ghobadi conducted a systematic mapping study, to understand what drives knowledge sharing in software development teams [68]. She analyzed 49 papers and extracted 44 knowledge sharing drivers that were classified in four major categories (people-related, structure-related, task-related, and technology related), which had seven sub-categories (diversity-related drivers, capability-related drivers, team perception drivers, team organization drivers, team organization drivers, organizational practices drivers, task-related drivers, and technology drivers). The analysis suggests that the most cited drivers are the ones related to team perception, which refers to values, perception, and attitudes that drive the team to share; organizational practices, that refers to communication networks, organizational norms and practices; and technology-related, referring to templates, tools, and methodologies.

In 2017, Zahedi et al. published their review [205] about knowledge sharing challenges and practices in global software development. They grouped challenges and practices into six categories: management, team structure, work processes/practices, team cognition, social attributes, and tools. Most of the reported challenges regard the work processes, followed by social attributes and technology/techniques categories. The study reveals that one of the significant challenges faced by global software development is to deal with tacit knowledge and that there are costs involved in knowledge sharing, such as travel between sites and keep documentation up to date, that may not appear on the project planning. Among the reported practices, temporary collocation is the most popular knowledge practice used by the companies. Other practices are also reported, such as groupware tools to enable frequent communication and build social ties. In general, the majority of companies tend to influence the team's social potential to share knowledge.

Comparison between this study and the existing related reviews

From the five SLR's that we identified, three of them investigate knowledge sharing/transfer in different contexts of software development. One of them explores knowledge management in software engineering. The closest SLR to our work is the one reported by Cabral et al. [203] that aims to investigate the main topics of knowledge management in ASD. On a high-level, it seems similar. However, in detail, several differences can be perceived between this study and the other SLRs, regarding the following aspects:

- *Research questions.* The research questions elaborated in this study are different from all previous SLR's, (see section 3.3). It may have an overlap of findings when comparing specific knowledge management processes such as knowledge sharing since the study published by Zahedi et al. [205] focus on challenges and practices of this KP. However, they consider this aspect in global software development. Our study explores all KPs in companies that adopt ASD.

- *Year.* Our search method, including automated search and snowballing procedure, was carried out on April 2018, while the closest to our study, reported by Cabral et al. [203], was performed in 2009, portraying a difference of eight years and 11 months. We consider this is a reasonable timespan, considering that software development area has rapidly changed over the years, e.g., more companies are introducing agile methods, and several of them are also adopting it in global software development.
- *Search string.* The search string used in this study differs since did not adopt composite words for knowledge management and combine the words for agile software development differently. We also used different databases to run the search strings for the automated search; more is included, for example, Web of Science, that has better coverage in journals of high impact [1].
- *Search method.* Our study adopted two search methods, automated search and snowballing, while the other studies adopted mainly automated search. The automated search was used to find the start set for the snowballing procedure.
- *Inclusion and exclusion criteria.* We defined different inclusion/exclusion criteria, particularly the one related to empirical studies, through which we could aggregate different findings.
- *Quality assessment.* Since we were looking for empirical studies, we adopted the Rigor and Relevance model [89] to classify and evaluate the studies based on the rubrics for rigor and relevance to better judge the empirical level of the studies selected for analysis in our study.

Most importantly, to the best of our knowledge, this is the first study that focuses on KM processes in ASD and their association with KM strategies. Previous studies target different aspects of KM in ASD. Therefore, we believe this review contributes to a better understanding of how companies that adopt ASD manage the knowledge asset. These findings also highlight the role of an integrated approach to KM that consider not only the project level, but how these practices should be related to the company's overall goals.

2.3 Research methodology

We applied SLR to synthesize the empirical evidence regarding KM in ASD. In the subsections that follow we present the subsequent steps: protocol development, search process and study selection, quality assessment data extraction, and synthesis.

2.3.1 Protocol development

We followed the guidelines proposed by Kitchenham [100] to structure the research question, search strategy, study selection, exclusion criteria, data extraction, and synthesis. Besides, we applied the Rigor and Industrial Relevance Model proposed by Ivarsson and Gorschek [89] to perform the quality assessment of the primary studies.

2.3.2 Research question

We defined the research questions:

RQ1: How do knowledge management strategies promote knowledge processes in ASD?

RQ2: To what degree have knowledge management strategies been validated in industrial settings?

RQ3: How have knowledge management practices been distributed in companies that adopted ASD?

Motivation: Previous studies have focused in specific KPs, e.g., knowledge sharing [68, 205], and related personalization strategies with knowledge sharing practices. However, the rest of the KPs (creation, storage, and application) remains uncovered regarding its practices and their relation to the KM strategy. As a consequence, our motivation relies on the need to gain comprehension about how software development companies adopting ASD implement KM strategies through practices that promote the KPs in the organizational layers, illustrated in the conceptual classification framework (see section 2.5).

2.3.3 Search process and study selection

The search process followed two approaches. According to Kitchenhan et al., [100] and Wohlin [199], combining different approaches is a way of having the best possible literature coverage. We applied automated search and snowballing to select the primary studies.

The definition of the start set for the snowballing [199] procedure was carried out with the help of a database search. We searched for essential keywords and common terms, synonyms, and abbreviations in previous literature reviews [18, 59, 203]. Keywords related to agile software development were derived from

Table 2.1: Keywords for automated search.

Agile Software Development - (ASD)	Knowledge Management - (KM)
1 agile AND software	11 knowledge
2 “extreme programming”	12 learn*
3 xp AND software	
4 scrum AND software	
5 crystal AND software AND (clear OR orange OR red OR blue)	
6 dsdm AND software	
7 “feature driven development”	
8 fdd AND software	
9 lean AND software	
10 “dynamic system development method”	

Dybå and Dingsøyr [59], while keywords related to knowledge processes were not limited to composite words, for example, "knowledge creation" or "knowledge sharing". We decided to use only knowledge and learn variations (e.g., learning), due to the lack of information about the existence of studies that did not use combined words. By doing so, we reduced the precision of the search string, elevating the number of papers found (4951). Although, adopting this strategy we reduced the risk of missing relevant papers.

We combined all relevant keywords (refer to table 2.1) using Boolean operators, and the search string result was:

(1 OR 2 OR 3 OR 4 OR 5 OR 6 OR 7 OR 8 OR 9 OR 10) AND (11 OR 12)

The search string execution and paper selection and screening were conducted in April 2018. We executed our search string in Engineering Village that focus on engineering databases; Scopus, that is considered the most extensive citation and abstract databases; and ISI Web of Science, that has lower coverage than Scopus, but the journals covered has a higher impact [1].

Automated search

Fig. 2.1 presents the start set identification process (A1) on the databases (Scopus, Web of Science and Engineering Village) search was 4951. The paper's metadata retrieved from databases were stored using Zotero. A total of 1019 duplicates were removed, which left 3572 papers for further analysis.

At stage 2, the first author excluded papers not written in English and studies that did not undergo peer-review, resulting in 3222 candidates. On the

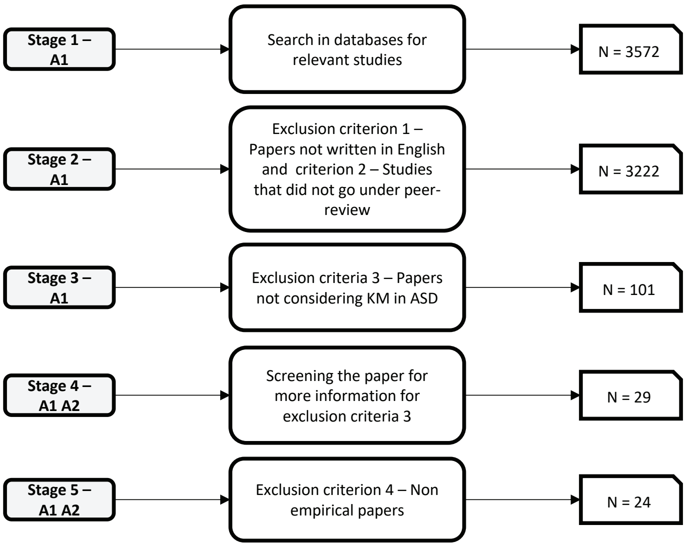

Stage 1 – A1
Search in databases for relevant studies
N = 3572
Stage 2 – A1
Exclusion criterion 1 – Papers not written in English and criterion 2 – Studies that did not go under peer-review
N = 3222
Stage 3 – A1
Exclusion criteria 3 – Papers not considering KM in ASD
N = 101
Stage 4 – A1 A2
Screening the paper for more information for exclusion criteria 3
N = 29
Stage 5 – A1 A2
Exclusion criterion 4 – Non empirical papers
N = 24

Figure 2.1: Automated search for start set selection.

third stage of the selection process, the first author excluded the papers that do not consider KM in ASD (exclusion criterion 3), resulting in 101 papers.

Several papers did not present enough information that allowed us to exclude them by title or abstract. Because of that, the first author, together with the second author (Stage 4 – A1 A2) screened the remaining 101 papers and excluded 71 papers based on the third criterion.

At stage 5, the first and the second authors individually excluded papers based on criterion 4 – Non-empirical papers. The number of papers resulted from this stage was 24. In this stage, we calculated the Cohen's Kappa coefficient of agreement [40] to measure the degree of agreement between the researchers. The coefficient value was 0.87, which represents strong agreement [107].

Snowballing iteration 1

We performed backward and forward snowballing on the remaining 24 papers. Fig. 2.2 shows the process and the number of papers candidate for selection. We collected 2043 papers in the first snowballing iteration.

We first removed all papers that we already examined on the automated search. After that, by reading the titles and occasionally the abstracts, the first author selected 35 candidate papers. The first and the second authors individually screened the papers applying the third exclusion criteria, as mentioned earlier, to the 35 candidate papers. The Cohen's Kappa coefficient of agreement in this stage was 0.18, indicating a slight strength of agreement.

In several cases, it was not clear whether the context of the study was the agile environment, which explain the weak agreement. Due to the lack of information and to not make any assumptions, we decided to remove those papers. We run the selection once more, and in this round, the Kappa was 0.89, the strength of agreement almost perfect. Five papers were included and used in the first snowballing iteration.

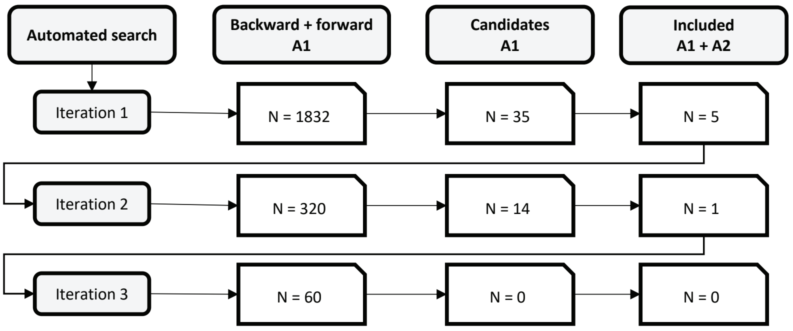


Figure 2.2: Snowballing iterations.

Snowballing iteration 2

In the second snowballing iteration (see Fig. 2.2), 320 papers were collected to analysis and 14 papers were considered as candidates. The first (A1) and the second (A2) authors individually evaluated the 14 papers using the exclusion criteria. For this step, the Kappa coefficient was 0.46, with moderate strength of agreement. One paper was included and used in the next iteration.

Snowballing iteration 3

For this iteration, we collected 60 papers, and none of them selected as candidates, concluding the snowballing part of the literature review. Overall, eight additional papers were included from the snowballing iterations, resulting in 32 papers to have their quality assessed before the data extraction begin.

Table 2.2: Data collection of primary studies.

Data collected	Research questions
Author, abstract, title, year of publication and type of article (conference, journal or workshop paper)	Demographic attributes for studies overview
Data collection method	Research design attributes
Rubrics related to rigor (context, study design description, and validity)	Study design attributes, RQ2
Rubrics related to relevance (research design, subjects, context, and scale)	Study design attributes, RQ2
Organizational layer of investigation	RQ1, RQ3
KM practices	RQ1, RQ3
Knowledge Processes	RQ1

2.3.4 Quality assessment

We applied the quality assessment model based on their rigor and relevance scores proposed by Ivarsson and Gorschek [89]. Rigor aspects are related to three rubrics: context, study design, and validity. They were evaluated according to the respective scores, 0.1, and 0.5. Relevance aspects have four rubrics, which are: context, research method, user/subject, and scale. The scores for these rubrics can be 0 or 1. The paper's scoring was performed by the first and the second author. To validate this process both authors individually scored the papers and then they discussed the results of each one of them. We performed this discussion due to the possibility of having different perception about the model's rubrics and consequently the scoring process. After the discussion, the authors agreed on the final score for each paper.

2.3.5 Data Extraction and synthesis

The first author used a pre-defined spreadsheet for data extraction. Table 2.2 shows the data items collected.

We used quantitative analysis and descriptive statistic visualizations to explore and summarize the studies.

To answer the research questions, we applied narrative synthesis [155]. We first classified the practices extracted from the primary studies in codification or personalization strategy, based on their definitions [78]. Second, we identified what KP these practices were enabling. The papers were clear on defining which KP they were aiming to investigate, e.g., knowledge sharing.

Finally, we examined in which organizational layer, described in the proposed framework (see Section 2.5) these practices are performed. Grouping the data into categories helped on exploring patterns and observe how KM practices are mapped in companies that adopted ASD.

2.4 Studies Overview and Descriptive Statistics

Fig. 2.3 depicts an overview of the 32 collected publications, including distribution through the years, publication venues, and type of method applied. We identified primary studies published in 2003 with a substantial increase in the number of publications after 2012 (see Fig. 3c).

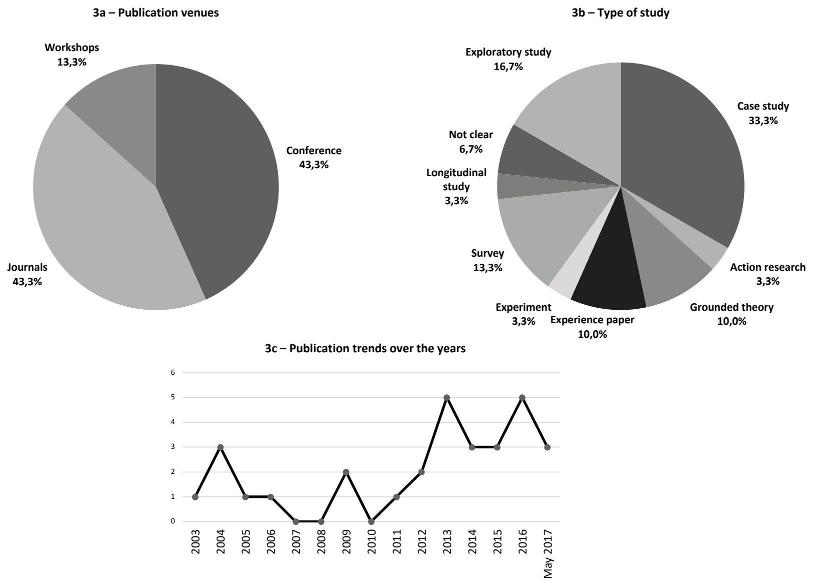


Figure 2.3: Studies overview

We found 14 papers published in Journals, 14 in conferences and 4 in workshops, see Fig. 3a. The venues focus mainly on software engineering, computer science and information systems, except for two journals, Knowledge and Process Management and VINE, that focus on knowledge and information management.

We used the classification proposed by Robson and McCartan [159] to aggregate the studies according to their research methods (Fig. 3b). Case studies represent 34,4% (P1, P2, P3, P9, P10, P12, P18, P19, P26, P30, P31). Five exploratory studies were reported (P11, P20, P21, P22, P28), which had mainly interviews as data collection technique. Four studies report surveys (P13, P17, P23, P25, P32). Grounded theory (P5, P8, P24) and experience paper (P6, P15, P16) had the same amount of studies, representing each of them 9,4% of the studies. We also identified one action research (P4), one experiment (P7) and one longitudinal study (P14). Besides, we found two papers that did not indicate the applied research method (P27 and P29).

Overall, we observe that most papers present descriptive studies, which suggests a possible difficulty in conducting studies that require repeated observations along the time, e.g., longitudinal studies. Particularly in KM, real context experiments are difficult to conduct due to the complexity of setting up its environment, time for observation and changes in companies' routine. This possibly explains why we found only one experiment (P7), and it was conducted with students. Action research also demands time for progressive reflection with individuals within organizations, which aim to formulate solutions to companies' problems and to test them.

2.4.1 Quality assessment

To classify the papers based on the final score (refer to Section 3.4), we used the categories proposed by Munir et al. [136] to generate a grid of combinations. To score the papers, we applied the scale proposed by Ivarsson and Gorschek [89]. The four categories are:

- A – studies with high relevance and high rigor ($1{,}5 < \text{rigor} \leq 3$ / $2 < \text{relevance} \leq 4$);
- B1 – studies with high relevance and low rigor ($0 \leq \text{rigor} \leq 1{,}5$ / $2 < \text{relevance} \leq 4$);
- B2 – studies with low relevance and high rigor ($1{,}5 < \text{rigor} \leq 3$ / $0 \leq \text{relevance} \leq 2$);
- C – studies with low relevance and low rigor ($0 \leq \text{rigor} \leq 1{,}5$ / $0 \leq \text{relevance} \leq 2$)

The relevance aspect relates to the environment where the studies' results were obtained, more specifically, to the practical environment, industrial context, real applications and applied research method. Rigor concerns the scientific aspects of the reported research.

Fig. 2.4 depicts the distribution of the quality assessment scores. 16 papers were categorized in the B1 area (low rigor and high relevance). It suggests that relevant studies were conducted, but they should have been conducted with higher rigor to increase reliability and validity. One paper appeared in category B2 (P7), with high rigor and low relevance. In category C, three papers present low rigor and low relevance (P27, P4, P15). Category A is the second in the number of papers, with 12 (P8, P14, P19, P20, P21, P22, P23, P24, P25, P26, P28, P31), representing the papers with high rigor and high relevance.

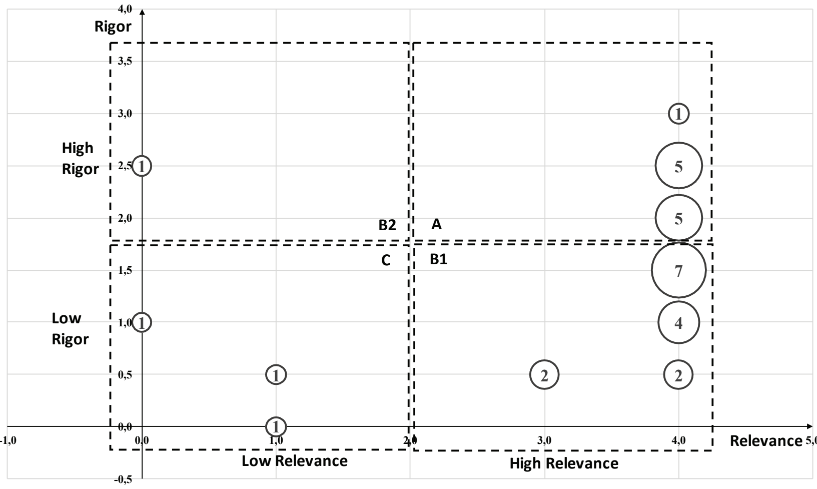


Figure 2.4: Quality assessment

Surprisingly, 59,3% of the studies were classified with low rigor regarding context, study design, and validity evaluation. When authors fail at describing research method and data analysis, reviewers and readers might misinterpret the research [50]. For practitioners, it is hard to analyze the studies' findings, since they might not be able to compare their context with the ones that describe poorly their context. Therefore, it is not possible to know if the practices adopted to manage knowledge are accurate and valid enough for practitioners' use. For researchers, the implications relate to how reproducible these studies are and how substantial is the evidence for future work.

Regarding relevance, approximately 87,5% of the studies present high relevance. To this review, this result is significant since we are gathering findings from empirical studies in which industrial settings are more representative. On the other hand, 19 studies present low rigor, what weaken their reliability. They have been unsuccessful in reporting mainly the validity threats. The second rubric that present low scores was study design (see Appendix A), and third, the context description.

Knowledge is a complex and multifaceted theme; possible implications of low rigor on researching this theme could be the misinterpretation for both researchers and practitioners on what is considered knowledge by the companies. This understanding guides not only the research method to be used, but the companies' actions to manage individuals' knowledge. Increasing the studies rigor could contribute on filling this gap and provide stronger evidence that could support future works and more understanding for practitioners.

2.5 Results and Analysis

To aggregate and analyze the results, we created The Knowledge Management Strategy in Agile Software Development (KMS-ASD) conceptual framework to explain how the different concepts are related, see Fig. 2.5.

Framework conception

Software development companies are often affected by changes to the external environment, market and customers' requirements. As a result, arrangements tend to change from hierarchical to flat, aiming more social interaction within and between organizational layers (strategic, tactical and operational); and changes in how an organization creates value through knowledge, e.g., factory model organization tends to have few knowledge-intensive functions, such as research and development departments [50]. Depending on the arrangements, the distance between the organizational layers may differ. With long distances, the translation of organization goals and requirements becomes a challenge, and may hinder its assimilation across the layers [17]. In the opposite direction, the communication from the lower to the other levels is also hampered.

The arrangements may also be different within the organizational layers, e.g., ASD companies have self-organized and greatly autonomous teams that rely on informal communication. Small to medium organizations, tend to have a flat structure of teams, which facilitates the implementation of KM strategies. Large organizations are often distributed, resulting in a more significant distance between the tactical and operational layers and greater team isolation.

Software development activities do not occur only in the bottom layer of a company, they relate to the other layers, e.g., executives, product management or requirements engineering [61, 113, 188, 193]. In the development process, the value created to the customers is supported by the organizational knowledge [103, 156]; which is built by knowledge processes within companies' routines.

The Knowledge Management Strategy in Agile Software Development (KMS-ASD) framework (refer to fig. 2.5) envisions a high-level perspective of a software development company that has one or more software-intensive products and develops them through ASD processes. Vähäniitty and Rautianen [188] propose a five-level conceptual framework (business, product service, development portfolio, project, and iteration) that aims to link long-term product and business planning. We adapted their organizational levels by encapsulating business and product service as a *Strategic layer*; naming development portfolio as a *Product portfolio* layer; and, product and iteration as a *Project layer*. This decision was made to simplify the framework and align it with the organizational structure proposed by Mahesh and Suresh [125], that has knowledge as central asset of an organization.

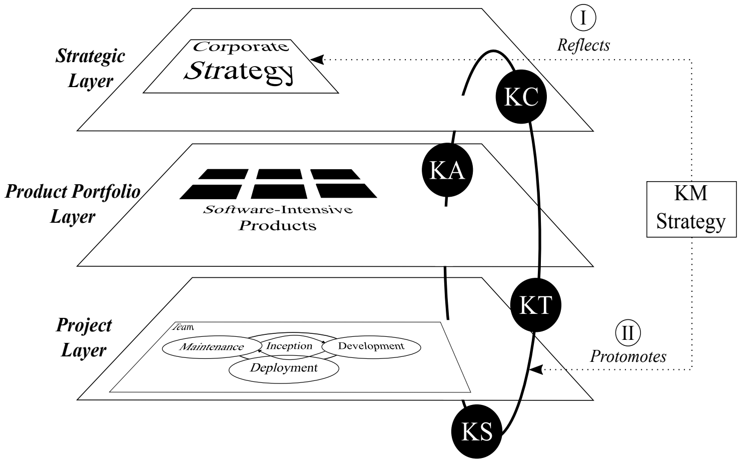


Figure 2.5: KMS - ASD

In the KMS-ASD framework, KM strategies **reflect** (I) the corporate strategy, which defines how the company should compete for obtaining or maintaining competitive advantage [35, 53]. The KM strategies **promote** (II) at least one of the KPs [3, 166] (KC - knowledge creation, KS – knowledge storage/retrieval, KT – knowledge transfer/sharing, and KA – knowledge application). The KPs take place within and between organizational layers. Codification and personalization strategies are usually combined; however, the distribution is not symmetrical, e.g., the mix could be, respectively, 80/20 for codification and personalization [78].

In the *Strategic layer*, the defined corporate strategy for obtaining competitive advantage is one input to define KM strategies [23]. The corporate strategy is discussed in terms of strategic positioning, the business model, current capabilities and required investments for new technologies and knowledge. The strategic level discussions also include how a company is going to achieve and sustain competitive advantage of market position.

The *Product portfolio* layer establishes the connection between strategic decisions and project levels. The strategic decision regarding the business, market positioning and the portfolio of product are taken here. The decisions about how to allocate the companies' resources into developing various products and what competences and knowledge are needed are also taken. In this level, personalization strategies that promote knowledge creation (KC) could be, for example, more aligned to the conception of a new product or product portfolio and their interaction with currently offered products or other services in an ecosystem. The product offering is developed concerning high level functionality and the product delivery strategy is agreed.

In the *Project layer*, the development iterations are planned according to the decisions made on the middle-level of the company [188]. Here, KM strategies are focused on practices that improve socialization between individuals inside and cross-teams. The less documentation approach to ASD foresee more interaction and, consequently, individuals disseminate more tacit knowledge.

The right balance of KM strategies could help knowledge storing and sharing from individuals throughout the company [161]. KM strategies may also focus on crossing layers. Launching new software-intensive products requires both technical and domain knowledge. The faster the new knowledge is spread and internalized, fewer delays may happen, and quality may be increased [161]. For companies that develop software connected or not to other products, this integration between KM strategies and the different levels could result in a successful competitive strategy.

In the following subsections, we list the KM practices reported in the primary studies regarding the KM strategy they implement *(RQ2)* and map them into the organizational layers. In each layer, we classify the practices regarding the type of KM strategy they implement, and which KP they promote *(RQ1)*.

2.5.1 Strategic layer

Fig. 2.6 depicts the distribution of KM practices *(RQ3)* that implement both personalization and codification strategies in KMS-ASD.

We found no primary study that took into consideration the strategic layer regarding KM. A possible explanation for the absence of KM practices in this level could be that the performed practices are informal and not directly connected to the corporate strategy or goals. Practices in the strategic layer could assist in discovering new requirements to create new products or improve existing ones, oriented by the organizations' vision.

Knowledge is the leading resource for software development organizations and the integration among the different layers of the organization is essential. The reason for that is related to how the organizations create value. To Mahesh and Suresh [125], a flexible organization is recognized as an enabler of a free transit of knowledge by creating a social structure that systematically integrates knowledge from different layers.

Agile software development is suitable for this type of organizational structure due to the low level of formality in its development processes. Also, the encouragement of social interaction facilitates promoting the KPs, which might help ASD teams to rapidly adapt to changes not only in features, but in market demand as well. We believe that KM practices could be applied in ASD contexts and help to integrate knowledge from all layers and to support realizing the corporate strategy. For example, if a company aim to be leader in innovation, KM practices that promote knowledge creation might help on developing new products.

However, based on our findings, it seems that agile teams tend to be isolated from the rest of the organizational layers, which means that KM practices could work well inside and across teams, but not integrated to the rest of the company. In a recent study [6], developers pointed that after agile adoption, they lost the "big picture" of the product because they are more focused on features do be developed. This finding could explain the lack of integration of KM practices to the other layers.

We suggest that future studies investigate how different organizational structures, e.g., flat and factory model, impact the KM practices integration to the different organizational layers of companies that adopt ASD. Moreover, use KMS-ASD framework to explain how strategies and practices are created and integrated to the strategic layer, mainly how they contribute to the organizations' competitive strategy.

The lack of integration may hamper the process of measuring the efficacy of the practices, due to impossibility of traceability of the practices in the project layer to the competitive strategy.

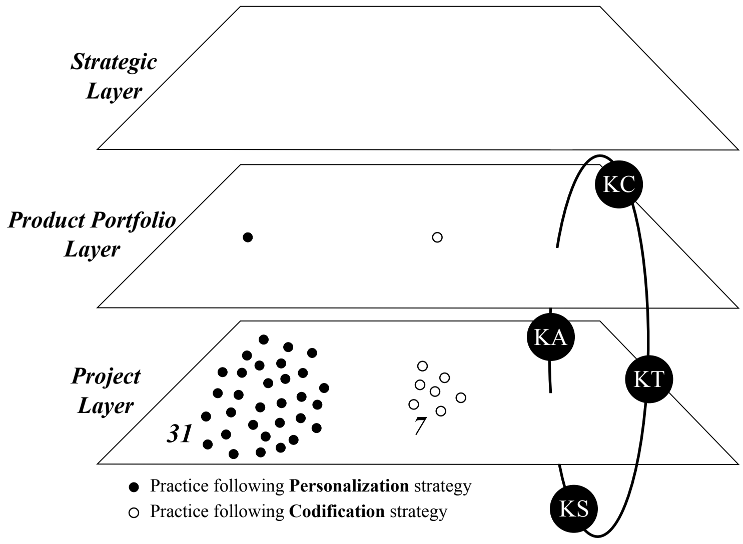

Strategic Layer
Product Portfolio Layer
Project Layer
31
7
KC
KA
KT
KS
● Practice following Personalization strategy
○ Practice following Codification strategy

Figure 2.6: Distribution of KM practices in organizational layers through KMS-ASD

2.5.2 Product portfolio layer

We identified two KM practices in the product portfolio layer: one implementing personalization strategy, and other implementing codification strategy.

As a personalization strategy, the "*marathon of innovation*" is a practice reported by Santos et al. (P24) that promotes knowledge creation. The company used its infrastructure to promote interaction among members from different teams and hierarchy levels to work together to stimulate innovation by developing new ideas. Individuals could, as a reward, dedicate time to maintain and grow the projects originated from their ideas.

The new ideas generation is a relevant topic and intrinsically related to KC. Santos et al. (P24) point out the relevance of the customer participation and the role of companies in enabling customer participation. Customers play a key role in ASD since they provide information about requirements and contribute to socializing their needs and experience, which may trigger the KC process to result in new solutions.

Regarding the codification strategy, *wiki-based tools are used to transfer knowledge between different departments.* This practice was identified in one study (P6), and allows the connection between different business units, keeping knowledge and information updated. The authors also point out that the tool was intensively used by the developers to store the product documentation and knowledge considered valuable by them.

Both practices focus innovation and knowledge sharing between product portfolio and project layer. To establish a connection beyond project barriers, patterns of communication and knowledge transfer are required. Strode et al. [180] identified in their study that companies use "boundary spanning artefacts" to enable this communication.

This type of artefact could be helpful on propagating knowledge through organizational layers. However, one main challenge remains unaddressed regarding this aspect, which is to know what knowledge the organization needs to share on what level and across the levels.

2.5.3 Project layer

We identified 40 KM practices that aim attention at the project layer. Regarding personalization strategies, the primary studies report practices that promote KC, KT and KA processes. Codification strategy gathers seven practices, they promote KS, KT and KA processes.

Personalization strategies

The KM practices that implement personalization strategies are divided into the following knowledge processes: Seven in knowledge creation; 25 in knowledge transfer/sharing; and one in knowledge application.

Knowledge creation process. Knowledge is created through KM practices that stimulate socialization of tacit knowledge. Three of the reviewed studies (P5, P22, P1) report KM practices for KC. All three studies have high relevance, but studies P1 and P5 have low scores for study design, discussion about validity threats, and context description. It hampers the comparison to other studies' results and the understanding about on other similar context KC practices could be suitable; for instance, the comparison between P5 and P22 regarding the context where communities of practices are applied.

Bahli and Zeid (P1) argue that the software development process adopted intervene on KC. After conducting an empirical study in industry, the authors concluded that the *adoption of extreme programming* facilitated KC.

Formal and informal learning practices reported in one primary study are applied to stimulate knowledge creation during project execution (P5). According to Dorairaj et al. (P5), these practices aim to improve team members' skills by upgrading their technical and management knowledge. *Communities of practices* (P5, P22), *pair programming* (P22), and *innovation boards* (P22) can be used to stimulate social interaction between the team members during software development. The *customer interaction* is a KM practice reported as a relevant source of KC (P5, P22). To Razzak and Ahmed (P22), daily customer involvement throughout the development cycles allow team members to discuss issues and create knowledge through continuous feedback.

Knowledge transfer process. 11 primary studies report the application of different communication channels that stimulate social interaction (P2, P5, P12, P13, P16, P18, P20, P22, P24, P26, P29). In distributed teams, this interaction is even more challenging. To minimize the distance and time zone differences, *information and communication tools*, such as video conferencing, chat rooms and telephones are adopted (P29, P20, P26). Wendling et al. (P29) argue that technology is essential to connect distributed teams. However, it does not replace face-to-face interaction. *Moving members between sites* for a short time complements this practice. Boden and Avram (P2) found out that giving the opportunity for *team members to work together for long periods* in a distributed context, facilitates KT between members from different sites.

Collocating teams tend to reduce the demand for information and communication tools. The focus changes to promote a friendly environment that encourages KT. *Communities of practices, workshops, consultancy, frequent meetings, and informal gatherings* are practices used to share knowledge among individuals (P2, P5, P12, P13, P16, P20, P22, P24, P26, P31). To Karlsen et al. (P12), the interaction during the meetings promotes discussions about different subjects, tasks, solutions, and estimations. This practice allows team members to share knowledge to build solutions together and discuss their progress and goals. In a case study at Ericsson, Šmite et al. (P26) found that communities of practices contribute to develop and increase the knowledge network by the frequent communication between teams. Santos et al. (P25) report one practice called *brainwriting*, a practice similar to brainstorming. In this case, the ideas were written in papers. According to the authors, a brainwriting session focus on reusing knowledge for problem-solving and on innovating by creating new concepts.

The physical working structure is a concern reported in one primary study. Santos et al. (P24) explain that the physical workspace affects KT effectiveness. Walls that separate offices act as barriers for KT. For them, *open workspaces* that integrate people and furniture that facilitate working together (e.g., tables for pair programming) provide closeness between members.

Companies adapt the software development process and team configuration to achieve team cohesion and better structure to share tacit knowledge (P5, P8, P12, P18). Ramesh et al. (P18) point that the time established for the development cycles depends on the complexity of the functionalities that the team is working. Because of that, *the development cycles are short, but not time-boxed*. In the studied companies, *the length of the sprints was changed*, aiming to achieve more interaction for KT. The authors also point that *keeping the same team during the whole project* is a practice that benefits team cohesion and KT as well.

Besides the adopted development process, the team configuration also undergoes adjustments. Two studies (P5, P12) reported the adoption of *cross-teams* configuration. To Karlsen et al. (P12), tacit knowledge transfer may be facilitated when the team is composed of individuals that assume different roles and have diverse backgrounds. Besides, an increase of motivation is also observed as result of cross-teams' configuration.

Practices that increase trust between individuals and long-term collaboration have a positive effect on KT in agile teams (P2, P8, P16). To Moe et al. (P16), *conceive and discuss product goals collectively*, mainly when the teams' members are gathered, induced the members to understand that they need to trust each other and share knowledge to achieve the goals. This practice is also pointed by Ghobadi and Mathiassen (P8) as valuable for knowledge sharing.

The primary studies report practices that aim to leverage individuals´ knowledge through interaction, resulting in technical knowledge sharing. Studies (P5, P22, P24) report that companies promote *formal or informal technical discussion sessions* to discuss technical solutions. In Santos et al. (P24), one company promote *challenge activities* support learning through interaction, aiming to share and improve programming skills.

Pair programming is reported as a beneficial practice in both collocated and distributed teams. The informal conversation during the development offers a good opportunity for the team members to share their knowledge (P7). In an empirical study, Ghobadi and Mathiassen (P8) found that when there is a lack of understanding about the business domain, *to engage experienced and motivated people* could help to share knowledge. Another practice that aims to level up knowledge between individual is the *rotation of members*, which is applied in collocated or distributed teams (P5, P12, P20, P22, P24). The rotation occurs by changing the individuals' roles from one team to another or rotating team member between sites when the company has distributed development. In

Santos et al. (P24), most of the studied companies adopt rotation of members when they need to structure new teams.

The relation of customer with the teams is explored in two studies (P5, P8). In Ghobadi and Mathiassen (P8), *leveraging a good relationship with the client* and *organizing training sessions* with the client are practices that provide opportunities to transfer knowledge in both ways. Dorairaj et al. (P5) suggest that having an *on-site customer* during the whole development helps on providing not only feedback but also collaborating throughout the iterations.

The concept of high-quality knowledge sharing is introduced by Ghobadi et al. (P7), and is related to how useful is the knowledge transferred to the software development activities. In an experiment, they explore how *coopetitive reward structures* influence the high-quality knowledge. The authors claim that on one hand, cooperation leads to better performance; on the other, competition generates more analytical thinking. By combining them, they found that there is a direct influence on the level of high-quality knowledge.

Six of the studies (P8, P20, P22, P24, P25, P26) received high rigor and relevance scores. In this case, the findings regarding the practices utilized for KT may be considered more consistent, providing better support for replication and practical relevance.

Seven studies (P2, P5, P12, P13, P16, P18, and P29) received high relevance but low rigor scores. Validity threats discussion and study design have the lowest scores. The implication of those scores for KT practices is the impossibility of establishing an appropriate comparison between the contexts. This fact may be more critical for distributed environments, since the contexts may differ significantly. Effectively describing the context where KM takes place may help in comparison and understanding about the frequency of social interaction, goals for transferring knowledge, people involved, type of knowledge, and transfer success rate.

Understand how each company perceive the concept of knowledge is crucial for the reproducibility of the KT practices, which is not successfully reported in the primary studies. The lack of rigor could imply in misinterpretation for researchers and subjects about the concepts and practices goals. Particularly, P7 is evaluated with high rigor, but with low relevance, mainly because of using students as subjects in the experiment. Besides, the applications used for the experiment are unrealistic, do not represent real industry context.

Knowledge application process. This process expresses how the knowledge is transformed into competitive advantage of a company. We found only one study that reports one practice that assists KA in the project layer. Dorairaj et al. (P5) report that the *sprint review* practice in Scrum helps to identify areas that need improvement for the next sprint and helps to create favorable circumstances for new ideas generation.

We could not find practices enabling KS. One possible explanation is that practices to store and retrieval knowledge are highly dependent on explicit knowledge, which is not the objective of the personalization strategy.

Codification strategies

We found seven KM practices that follow codification strategies: three of them are related to KS, three to KT and one to KA. According to Alavi and Leidner [3], KM practices that implement codification strategies aim to store knowledge through documentation or databases. The goals are to share explicit knowledge, to build up experience, and to contribute with new experience e knowledge.

In **knowledge transfer process**, we found two studies (P4, P8) that report documentation of project experiences. Dingsøyr and Hanssen (P4) propose a *lightweight approach to post-mortem reviews*. The teams discuss and document good and bad experiences from the projects and involve stakeholder in the review process. According to Ghobadi and Mathiassen (P8), *documented experience* supports planning future projects by observing inadequate planning performed in the past.

Information and communication technologies are widely adopted by companies studied in the primary studies (P5, P12, P29, P13, P22). Companies keep track of technical details of the software, project documents and meetings to share with the teams. Wiki-based tools are the most used, together with the tools for knowledge transfer, such as JIRA, ScrumWorks, Confluence, Hudson, Yammer, Github, electronic boards and Redmine. One study reports *visual prototyping* as a practice to share knowledge (P20); however, we could not find any detail about how this practice is conducted.

We found three studies in the **knowledge storage process** area (P3, P5, P18, P32). Ramesh et al. (P18) report the case of a company that developed a *database* that their teams use to store different types of information and knowledge, to report issues and get feedback on solutions. The database is also reported by Gervigny and Nagowah (P32) to store lessons learned and good work practices.

KM systems are used to store content from technical presentations, concepts and technical expertise for further consideration by team members. Software details are stored in the code itself (P5).

Chau and Maurer (P3) developed an *online tool to store knowledge* based on the wiki technology. It is associated with a database capable of storing structured and unstructured information. Besides, the tool aims to support the experience factory approach through structured knowledge and experiences of learning, which is similar to communities of practice.

We found one codification practice in the **knowledge application process** area. Dorairaj et al. (P5) studied a company that documents knowledge from projects in a wiki-based tool, aiming to *reuse* or adapts in future projects in similar context. The teams document knowledge related to issues and solutions, adding detailed explanations.

We were not able to find practices that promote the KC process. One possible explanation is the nature of the KC process, that is about externalizing tacit knowledge to combine it with other explicit knowledge. To do that, tacit knowledge requires being externalized to be stored.

Two papers received low rigor and relevance scores (P3, P4). In P4, despite the use of real industry projects, they use students' subjects; this scenario does not represent a real industry environment. In P3, the practice of using a tool is undermined because of the lack rigor on validating the tool in a real industry context. Therefore, these practices should be considered carefully by practitioners and researchers for two main reasons: lack of context description and industry representation.

2.6 Discussion and Research Gaps

In the primary studies, we observe that companies combine both personalization and codification strategies. 81% of the KM practices implement personalization (promote social interaction between team members) while 19% of the practices implement codification. This means that the practices of codification strategies act as support for the personalization strategies.

Regarding the KPs promoted by the KM strategies, we observe a tendency on the **KT** practices to focus on tasks and problem solutions by enhancing face-to-face or digital communication. However, the primary studies do not explicitly explain what knowledge is transferred. Using informal communication creates a risk of losing knowledge or not transferring it. The result is that knowledge gets localized inside a few individual's minds instead of being propagated.

We found only three empirical studies that focus on **KC**. This indicates that more in-depth studies are required to understand why ASD needs to promote KC and what knowledge they aim to create. KC is not a trivial process to investigate, and probably the reason of the low number of primary studies found is because it requires substantial time from researchers to observe how the knowledge was created, shared and resulted in a new product, process improvement, solutions, among other outcomes.

Curiously, the **KA** process is the least investigated by the reported studies. The organizational competitiveness may be affected by the ability to effectively apply knowledge [164]. Future studies focusing on ASD could approach where knowledge is applied and how it has become a solution or a product.

In **KS**, we notice that software companies still struggle to know what knowledge they should store. More important than having a tool, is to know what to store regarding information or knowledge. Additionally, researchers should explore how one can adapt storage and retrieval activities in ASD, more specifically, when these activities occur in the project and what are the guidelines the teams should follow to perform them.

To summarize, it is surprising that the identified papers do not discuss KM practices efficacy. Are these practices effective? A previous study conducted in 2008 also presented this concern [18]; however, our review reveals the same lack of studies regarding the efficacy of KM practices. We recommend to apply empirical and rigorous methods for investigating KM in ASD, such as case studies, followed by in-depth interviews and observational studies since the beginning of a project till it ends, or even longitudinal studies. By doing this, it is possible to trace the knowledge throughout the development process, its externalization, combination, and internalization inside/across the teams/company. Another viable research method is the ethnographic study, also suggested by Bjrnson and Dingsyr [18].

2.7 Threats to Validity

External validity remains a significant concern since we studied the intersection of ASD and KM. We consider that our findings are generalizable to companies that adopt ASD principles through the different methodologies in software development. However, a risk remains that there could exist studies in ASD or KM literature that could be considered relevant but do not explicitly consider both areas. Due to the systematic nature of this study, we consider this risk to be low.

Concerning internal validity, we address the five characteristics: study selection, subjectivity on quality assessment, misclassification of primary studies, no exclusion of papers with low rigor/relevance, and database selection.

In the **study selection** phase, disagreements may have led to the exclusion of relevant papers. To mitigate this risk, the two first authors independently applied the exclusion/inclusion criteria and measured the related agreement through the Cohen's Kappa coefficient [40].

The second characteristic is a degree of **subjectivity on quality assessment**. Although the model [89] used to evaluate the quality of the primary studies presents rubrics with clear description, understanding them and scoring each paper could introduce misjudgment on quality evaluation. In this case, the first and the second author performed this process together, discussing the model rubrics and how the primary studies satisfied them.

Referring to the **misclassification of primary studies**, the lack of proper description about KP and KM strategies described by the authors in some of the investigated primary studies could have jeopardized the reliability of our results. To mitigate that, we defined all theoretical concepts and terminology for both constructs, ASD and KM, and classified the papers during the synthesis base on these definitions.

Regarding the decision of **not excluding primary studies** with low rigor/relevance, it could have weakened the evidence of our findings. However, excluding these papers, we could miss relevant or rigorous findings, mainly because all of them report empirical studies. Since they already underwent peer-review, we believe that we keep a balance between volume and validity of our findings.

To avoid the risk of missing papers that are not indexed by the **selected databases**, we complemented the database search with the snowballing procedure. This approach helps reconsidering papers that were excluded by mistake on previous steps, since several of them emerge again during the procedure.

2.8 Conclusions and Implications

Comprehending KM concepts becomes more important as companies start to consider knowledge as one of the main resources for building products. Successful KM strategies implementation involves knowing what goals the company aim to achieve, and its relation to the corporate strategy. Thereafter, the KM strategy promote the KPs through practices in the different layers of a company.

By following a systematic procedure, we conclude:

- KM strategies in ASD promote mainly KT process with practices that stimulates social interaction to informally share tacit knowledge;
- The primary studies reported practices in product portfolio and project layers. Most of the practices (95%) are placed in the project layer and the

studies do not report about the connection of KM strategies and practices between the strategic layers and product portfolio or project layer;

- 81% of the reported KM practices implement personalization strategy, while 19% of the KM practices implement codification strategy (acting as a support for personalization).
- Personalization strategies promote KC by stimulating close interaction between team members and customers. Personalization strategies also promote KT by stimulating social interaction inside and between teams, modifying the agile method adopted to accommodate knowledge transfer practices, and customer participation; and promote KA process by using review sprint approach.
- Codification strategies support storing of the project documentation, learning materials, and meeting notes. KT process is supported by several documents, i.e., post-mortem reviews and prototyping. The codification strategies promote KS using databases and information systems, i.e., wiki; and promote KA by using the knowledge stored in wikis.

The results of this review have implications for both researchers and practitioners. For researchers, we identified several potential research gaps in Section 6. We summarize our indications for future research into three main implications: 1) to investigate how KM strategies are planned in ASD; 2) to explore how organizational structures (layers and size) impact KM strategies implementations in ASD context; and 3) How to define measures for KM practices' efficacy in ASD.

For researchers, the results could contribute to a better comprehension of how the strategies are related to the processes through KMS-ASD framework, and on which gaps future research should focus. For practitioners, this study offers several insights for KM practices in ASD context, depending on what KP they mainly want to promote. The combination of KM strategies shows that personalization strategies are predominant in ASD. The framework could guide practitioners in planning KM activities that support the corporate strategy at all organizational layers, considering what knowledge should be transferred from and to the different layers.

Chapter 3

The Role of Knowledge-Based Resources in Agile Software Development Contexts

This chapter is based on the following paper:

> Raquel Ouriques, Krzysztof Wnuk, Tony Gorschek, and Richard Berntsson Svensson. "The role of knowledge-based resources in Agile Software Development contexts" *Journal of Systems and Software*, 197, 111572, (2023).

3.1 Introduction

Software-intensive companies that have adopted Agile Software Development (ASD) are engaged in continuous assimilation of software changes by sharing, codifying, and transmitting knowledge to people inside and across teams. However, inefficient utilisation of knowledge resources (for example, caused by insufficient codification or informal communication) results in a significant knowledge and time wasted [168].

Whilst ASD places less emphasis on the role of traditional coordination mechanisms, for example, up-front planning and extensive documentation, this approach enjoys the necessary agility to better respond to market changes [180]. ASD consequently relies on self-organized teams that utilize informal communication and tacit knowledge [18] that is shared in an *ad hoc* manner among the team members [39, 161]. This process may take place in both co-located and distributed environments [70,131]. Although tacit knowledge is relevant for companies and crucial for innovation [139], it is only revealed when it is applied [73]. Such knowledge is acquired via experience and transferred among people. This can prove to be costly, slow, and uncertain [103].

Software development is undoubtedly a knowledge-intensive activity since it is often associated with complex and intangible social resources that are difficult to reproduce, even though they may lend a competitive advantage to a company [11, 71]. At the same time, the tacit knowledge that is possessed by a group of software developers cannot be owned by a company. Notwithstanding this, it should be appropriately managed and utilised because it is a key resource for any software company [176].

A company's competitive strategy is substantially dependent on how the company maximises its produced value by allocating its resources [11]. A Knowledge-Based Resource (KBR)— also referred to as a knowledge resource in this study — relates to the employees' intangible knowledge that is valuable to a company's competitive advantage. These resources appear as specific skills, including technical, creative, and collaborative skills. Note that collaborative skills are relevant to the integration and coordination of multidisciplinary teamwork [134]. For example, this skill may be realised in the ability to select which items from the backlog a team should prioritise, taking into account the broader context of the product's development.

Because the software value chain is knowledge-based [161], knowledge application in software production might be jeopardised by a lack of practices that are aimed at managing knowledge as a strategic resource. Therefore, it is essential to know how specific KBRs allow the company to enjoy a continuous state of readiness to respond to the market effectively. This is achieved by effecting internal changes and by offering customer value quickly [42, 134].

The concept of 'knowledge' has been previously explored in the Agile Software Development (ASD) literature and has thus been recognised as relevant support for the management of dependencies in coordination [129, 148]. However, these studies do not explain how knowledge constitutes a keystone for further analysis and further application for companies that use ASD. Although scholars recognise the importance of knowledge in software development activities [18, 161], there is a lack of relevant studies in the software engineering literature that seeks to explain the role of the knowledge resource in ASD environments. Miller and Shamsie identified this gap in 1996 and claimed that it was justified by the fact that software development is a turbulent environment, even though KBRs play a significant role in this environment.

Our present study addresses this gap in the literature by examining the role of knowledge-based resources in ASD. To this aim, we focus on the planning and coordination activities that practitioners use.

This paper contributes to the field by virtue of its being:

- An empirical investigation that identifies how KBRs are distributed in an ASD context;
- The development of the Knowledge-Push theory, which illustrates how KBRs boost the need for change in ASD environments;
- A discussion of the possible implications for future research and potential solutions for managing the KBRs.

This paper is organised as follows: In Section 3.2, we present a brief background to our study and related work. Section 3.3 describes the research method. In Section 3.4, we present the Knowledge-Push theory, along with a description of each category that the theory relates to. In Section 4.5, we discuss the implications that our current findings have for future research, and we comment on the practical implications that KM gives rise to in the context of ASD. In Section 3.6, we discuss the threats to the validity of our study. Finally, in Section 3.7, we present our concluding remarks.

3.2 Background and related work

This section contextualises our research topic by offering a brief overview of the characteristics of ASD and its contradictions regarding its benefits. We also provide a number of definitions of the concept of 'knowledge' that we have adopted in this study. Furthermore, this section provides an overview of the theoretical foundations of KBRs.

3.2.1 Agile software development

Flexibility in responding to change [197] is the focal point of Agile Software Development, an umbrella term that includes several methods and frameworks that inform software development practices [14,42,59,194]. ASD prioritises human factors, including the interaction between people and teamwork. Further, it focuses on delivering working software quickly and continuously updating it when new requirements need to be met, even late in the project [14].

Dybå and Dingsøyr [59] have examined the ASD literature and identified several contradictory findings regarding the benefits of adopting ASD methods. In general, the benefits that were enjoyed from the adoption of ASD practices were observed with respect to collaboration with the customer, cost estimations, and focus on the work. The authors also pointed out that some studies report on benefits and improvements with regards to Extreme Programming regarding productivity, while other scholars report on the opposite, for example, a lack of attention to architecture and design in the context of Extreme Programming. Close examination of Pair Programming also gives rise to divergent findings. Although some developers perceive that the adoption of this method speeds up the development process, others perceive it as inefficient.

In a recent study, Annosi et al. [8] observe that the time pressure that falls out from the implementation of the Scrum framework has negative implications for learning and innovation. Whilst software developers remain focused on specific tasks, this comes at the cost of losing the broader context in which the software product is situated Li et al. [121] also discuss the harmful effects that time pressure may have on a software development project.

The ability to deal with constant change is the central idea of ASD. Given this, we note that knowledge plays an essential role in this regard [161]. While ASD prioritises informal communication (due to the flexibility and reduced demands for documentation in ASD), it also largely relies on knowledge exchanged between people [39].

Previous literature on the subject has indicated that companies who adopt ASD rely on tacit knowledge to a large extent [18]. Furthermore, they rely on informal means of communication as they share this knowledge between people, in both co-located and distributed ASD contexts [52, 70, 131]. Although tacit knowledge is recognised as a relevant resource for companies and crucial for innovation [139], it can only be revealed when it is applied [73]. Such knowledge is acquired through practices and is transferred between people, a process which can be costly, slow, and uncertain when compared to the utilisation of technology to transfer knowledge [103].

3.2.2 Definition of relevant concepts relevant to 'knowledge'

In this subsection, we describe the concepts that are used in this study and the arguments we make regarding the concept of 'knowledge' that further function as the foundation for the theory that we develop in this paper.

It is relevant to differentiate knowledge from information so as to avoid misunderstanding and to prevent the interchangeable use of these terms. Information merely refers to processed data that lacks an interpretation within a specific context [3].

The nature of knowledge has been discussed extensively among philosophers [73]. However, a close examination of epistemology is not our primary purpose here. We thus assume a somewhat straightforward definition of the concept of 'knowledge' as "a collection of information that provides guidance [that is] based on individual cognitive processes" [3, 139].

Although this study considers the two most frequently cited types of knowledge, these types of knowledge are also widely debated, namely *explicit knowledge* and *tacit knowledge.* Explicit knowledge is systematised in formal language (for example, in manuals, specifications, and other documents) and in any documented format (e.g., graphics, designs, drawings). Whilst tacit knowledge is highly personal since it is rooted in people's actions, values, and routines. Consequently, it is difficult to formalise and systematise tacit knowledge in a formal language [141].

It is a challenge to manage tacit knowledge properly. Because of this, organisations use Knowledge Management (KM) as a tool to extract knowledge from people so that the organisation can enjoy ownership of this resource. We employ a definition that states that knowledge management is realised by an organisation's work to manage the workforce's knowledge through social processes, including the application of tools and techniques to manage the systematised knowledge [11, 73, 84].

This knowledge management work can occur through four knowledge processes [3, 46]:

- Creation - the combination of tacit knowledge for new knowledge generation;

- Codification/storage - a means for systematising and storing relevant knowledge as useful information;

- Transfer/sharing - the movement of both types of knowledge between people or groups; and
- Application - the use of knowledge knowledge to generate value.

3.2.3 Knowledge-based resources

According to a resource-based view of the firm, a company's competitive strategy is significantly dependent on how the company's management team maximises the produced value by allocating the company's resources. These resources are idiosyncratic and thus provide an inevitable heterogeneity to the company, as it operates within a market [11].

Resources that companies use can be property-based or knowledge-based. Property-based resources are owned by a company and are protected by legal rights, for example, by contracts and patents. Knowledge-based resources (KBRs) include that *which is defined as the employees' intangible knowledge that is valuable to a company's competitive advantage.* KBRs appear as specific skills, including technical, creative, and collaborative skills, which, in turn, are immediately relevant to the integration and coordination of multidisciplinary teamwork [134].

The KBRs receive more attention when they are involved in activities that require human interaction for production, inside and outside the company. For example, this may include team and customer collaboration [9, 169]. KBRs are also often used to develop property-based resources, for example, software documentation and process descriptions.

Strategic management is the name of the field within which most studies that focus on KBRs take place. Most of the empirical research in this field is focused on the industry domain level, while research into the lower levels, for example, the company unit or individual departments, remains scarce. Regarding conceptual studies, they attempt to generalise, even though the resources that they take into consideration may have different values depending on the context that is examined [134].

Previous research has provided evidence that KBRs positively contribute to return on sale, operating profit, and market share [134]. Moreover, other studies have also provided evidence that KBRs enhance different types of innovation that are related to (i) the coordination of activities, (ii) external partnerships, (iii) business processes and methods, and (iv) employees' skills. KBRs are also said to contribute to innovation with respect to service delivery methods, product innovation, and organisational innovation [138].

Kaya and Patton [97] have explored the relationship between KBRs and innovation performance in different industry domains. They analysed several knowledge-based resources, including the staff's technical expertise regarding management, customer service, marketing, the generation of new ideas, and product development. They also examined the relevance of staff commitment to the company. The results of their study reveal that knowledge-based resources significantly enhance innovation performance.

Different resources contribute to the performance and competitiveness of a company at different levels[11, 16]. To maximise the positive impact that these contributions can make, companies must know how valuable these resources are, why they are scarce, and in which circumstances they exist [5].

Although there is evidence demonstrating that KBRs contribute effectively to securing a competitive advantage from a general perspective, each industry domain possesses diverse KBRs. Consequently, companies are tasked to increase their competitiveness in different ways.

Several roles in the domain of ASD possess relevant knowledge that supports software development, for example

- **Product owners** - possess knowledge about prioritising items from a backlog, taking into account the larger context in which the product will be used.

- **Developers** - possess knowledge about how changes in the product can be incorporated, considering the balance between the need for fast delivery and architecture degradation.

- **Test lead** - possess knowledge about how many manual tests should be added and what parts of the product can be tested automatically.

This knowledge possessed by these different roles is critical for efficiently estimating the effort that is required to accommodate incoming requirements in software products. The ability to promptly respond to and assimilate change greatly depends on an ASD team's ability to employ and share this (primarily tacit) knowledge.

In 1996, Miller and Shamsie suggested that research should focus on clarifying the impact of knowledge-based resources in 'turbulent environments', including the software industry [134]. However, as far as we know, knowledge-based resources in software engineering or in ASD have not been addressed in the literature to date.

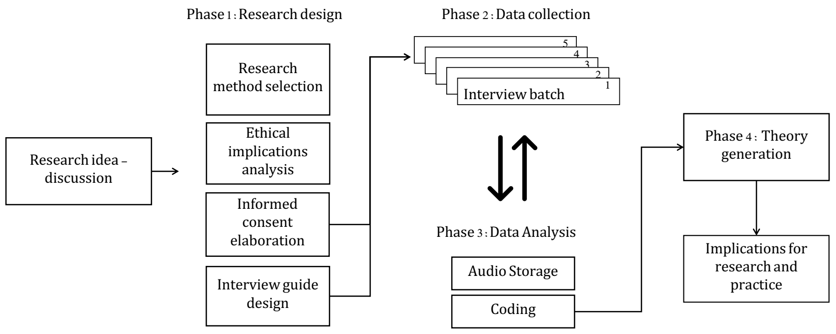

Phase 1: Research design
Phase 2: Data collection
Research idea – discussion
Research method selection
Ethical implications analysis
Informed consent elaboration
Interview guide design
5
4
3
2
1
Interview batch
Phase 3: Data Analysis
Audio Storage
Coding
Phase 4: Theory generation
Implications for research and practice

Figure 3.1: Research process overview

In an industry where knowledge is the main competitive resource, a lack of management practices with regards to knowledge may well jeopardise the application of knowledge in the production of goods and services. In the interest of advancing research into KBRs in the software industry, the present study adds to our understanding of how the KBRs are distributed and how they boost the need for change in ASD.

3.3 Research method

The goal of this study is to examine the role of KBRs in ASD closely. We are particularly interested in agile practices and the associated mechanisms that are applied to coordinate activities.

We based our Research Questions (RQ) on the findings and identified research gaps revealed by our previous systematic literature review [146, 148] and our discussions and brainstorming sessions with three companies. Two RQs are addressed below:

- RQ1: How are knowledge-based resources distributed in ASD environments?
- RQ2: How do companies utilise knowledge-based resources?

To answer these questions, we investigated the planning and coordinating activities that are used by practitioners in software development companies that have adopted ASD. We focused on these activities because coordination mechanisms stimulate the emergence of conditions where people can share their specialised knowledge with each other [73].

We conducted a qualitative study so as to efficiently capture data regarding the complex phenomena of human behaviour [85, 167].

We followed the principles of Grounded Theory as a method for theory generation [44, 45]. This included subscribing to a theory of social constructionism, a theory which notes that social phenomena are phenomena that are subject to constant change and are affected by human interaction [28]. Additionally, we adopted a theoretical perspective that recognises that knowledge is something that is socially constructed [139].

The data was collected via a series of semi-structured interviews. We examined the data by using open coding (text and audio) thereby identifying a number of relevant concepts. These concepts enabled us to explain the categories that were established in the data [167]. The method of analysis that is detailed by Corbin and Strauss [44, 45] allowed us to identify the knowledge-based resources and come to an understanding of the phenomena that lie behind the use of these resources in software development.

Figure 3.1 depicts the iterative research process that was used in our study. Each piece of data that was collected was analysed, which, in turn, gave rise to additional questions which we explored in the successive data collection sessions. We reached theoretical saturation, which means that the researchers have achieved consistency and representativeness of the generated concepts, and the data analysis advanced to the stage where we could engage in the creation of the explanatory categories.

To report on our findings, we followed the guidelines proposed by Stol et al. [179], which provide a framework for reporting on grounded theory in software engineering.

3.3.1 Phase 1 - Research design

In Phase 1, the first author created the initial version of the interview guide, and held discussions with the other authors to formulate the research questions. The questions were aimed at gathering information about knowledge resources directly relevant to coordination activities, including breaking down software development tasks.

It is important to note that, although an interview guide was available for use, the authors did not limit themselves by slavishly following it. Instead, they allowed the practitioners to freely express what they thought was relevant

Table 3.1: Description of the practitioners included in this study

Participant	Role	Education background	Years of experience	Company Domain	Agile method	Team size
P1	Architect	Computer Science	7	Telecommunication	SCRUM - Partially	6
P2	Program Manager	Computer Science	17	Telecommunication	SCRUM - Partially	6
P3	Design Leader	Computer Science	8	Telecommunication	SCRUM - Partially	8
P4	System Manager	Software Engineering	18	Telecommunication	SCRUM - Partially	5
P5	Product Owner	Software Engineering	3	Telecommunication	SCRUM - Partially	5
P6	Scrum Master	Computer Science	5	Management applications	SCRUM	7
P7	Software Engineer	Development and Analysis of systems	3	Consumer electronics	SCRUM - Partially	8
P8	Lead Architect	Computer Science	21	Consumer electronics	Kanban /SCRUM	11
P9	Product Manager	Engineering	12	Consumer electronics	Kanban /SCRUM	11
P10	Developer	Computer Science	27	Consumer electronics	Kanban /SCRUM	15
P11	Line Manager	Computer Science	17	Consumer electronics	Kanban /SCRUM	13
P12	Platform Maintainer	Computer Science	20	Consumer electronics	Kanban /SCRUM	10
P13	Senior Developer	Software Engineering	15	Consumer electronics	Kanban /SCRUM	15
P14	Senior Quality Engineer	Software Engineering	7	Consumer electronics	Kanban /SCRUM	10
P15	Scrum Master	Software Engineering	12	Telecommunication	SCRUM - Partially	7
P16	Team Lead	Engineering	15	Mobile communication	Kanban /SCRUM	6
P17	Manager	Acoustics Engineering	25	Mobile communication	Kanban /SCRUM	6
P18	Lead Engineer	Computer Applications	9	Mobile communication	Kanban /SCRUM	6

during the interviews and even deviate from the list of questions included in the interview guide.

The following subsections describe the sampling of practitioners, the method for data collection and execution, the data analysis, and the ethical aspects that we considered when we planned and executed this research project.

Sampling of practitioners

Practitioner selection was performed by means of a convenience sampling [110] that focused on participants whose activities were related to coordination. As we proceeded with our interviews, we applied a process of 'theoretical sampling' [44]. This entailed sampling practitioners who held different roles and responsibilities in the different phases of software development and at various organisational levels. We thus ensured that different parts of the software process were covered, and a state of theoretical saturation was achieved.

We intentionally sampled practitioners' roles in companies that (i) work with both co-located and distributed development; (ii) have software development as their primary business; and (iii) have software development as an activity that is complementary to their primary business. We collected data from 18 practitioners distributed across five companies (see Table 3.1).

This sampling strategy allowed us to adopt a broad perspective with regards to the different contexts where the companies employ agile practices. The practitioners who participated in this study work for companies that are located in Sweden and Brazil. The domain of these companies ranges from telecommunication, consumer electronics, mobile communication, and management applications. Except for two companies (one does business only in Brazil and another in Sweden), all of the companies conduct business activities in several countries. They are all classified as large companies (more than 250 employees), following the OECD - Organisation for Economia Co-operation and Development criteria [143].

Phase 2 - Data collection

The data for this study were collected by means of interviews. We developed a comprehensive interview guide; however, not all questions apply to all roles. Therefore, we asked the questions according to the role of each practitioner who was interviewed, be it product owner, scrum master, line manager, or project manager.

A few practitioners presented archive documents during the interviews, such as internal presentations displaying the distributions of the teams, the quality framework adopted, and organizational schema. However, the documents served as contextualization, facilitating our understanding of some of the topics discussed during the interviews.

To collect additional information from practitioners who had already been interviewed, we sent a follow-up questionnaire by e-mail. These follow-up questionnaires were adapted based on which further information or clarification we needed.

The interviews lasted between 30 and 60 minutes. Sixteen interviews took place on-site, one through Skype®, and one interview guide was sent by e-mail. The practitioner who responded by e-mail was given a one-week deadline to respond to the questions that were applicable to the practitioner's role at the company.

We collected the data in batches, which started in May of 2018. The first five interviews were conducted on-site and were analysed between May and June of 2018. In this first batch of interviews, we rephrased the questions that the practitioners reportedly misunderstood in the interview guide. In the subsequent interview guides, we evaluated what we could explore more substantially during the forthcoming interviews.

In July of 2018 (i.e., the second round of interviews), the first author performed an interview over Skype® and sent and received one interview guide by e-mail. In August, the first author conducted seven interviews (i.e., the third round of interviews). The data collection lasted until November of 2018, when we conducted the fourth round of interviews. This round included a total of four interviews.

We began to reach a state of theoretical saturation for most of the concepts used in this study during the third round of interviews. In the fourth round, the interviews did not bring any new concepts to light. However, they did confirm our previous findings. We thus decided that we had identified a sufficient number of concepts to build a solid theory.

Data storage

We stored two types of data (audio and text) in the MAXQDA software that can be used for qualitative and mixed methods research[1]. The software also allowed us to extract relevant concepts directly from the data source (audio and text).

Ethical concerns

Although our study did not process any sensitive data detailed in the Swedish Personal Data Act (1998:204) that would require an ethics board review, we followed the ethical considerations specified by the Swedish Research Council [183].

Regarding the data collection process, we paid attention to confidentiality by limiting access to the authors only and by not disclosing the participants' names, gender, or nationality. In the interview guide, we avoided asking questions that could be emotionally intense or cause psychological harm to the participants [4].

Before each interview session, the first author explained the purpose of the research project to the participant. This information was also stated in the informed consent form. We asked permission to conduct audio recordings during the interviews and explained to the participants who would have access to the data, how we were going to use it, and for how long we would keep the audio recordings. When we were sampling the participants, we focused on the roles that were relevant to our research in terms of the activities that the participants perform and how these activities were connected to coordination. The companies agreed to participate in the study due to their interest in the research results. We thus avoided any economic, legal, or power structure dependency between the authors and the participating companies that could influence the participants' responses or our analysis of their responses. Based on these criteria, the companies sampled the participants who were available to participate in an interview during the period that we visited the companies.

[1]Available at https://www.maxqda.com

Our ethical concerns regarding the data analysis included an awareness of potential stigmatisation or harm to specific populations. In this regard, we did not consider the gender, race, or minority status of any of the study's participants.

3.3.2 Phase 3 - Data analysis phase

The data collection process and the analysis of said data were interrelated [45]. The first author conducted the coding procedure in each stage of the analysis, which was carried out after each round of interviews. The preliminary results of the study were extensively discussed with the second author during the entire process of data collection and analysis.

For example, after we had analysed the first five interviews, we observed several re-occurring phenomena, thus indicating a future category. For the subsequent round of interviews, we observed new concepts but also re-occurring concepts with respect to the concepts identified during the first round of interviews.

Several new concepts and categories appeared in the data as we examined different company roles and industry contexts. Previously identified concepts were also strengthened. The categories were refined several times, and the concepts adjusted after each round of data collection and analysis. Dividing the data collection process into distinct rounds enabled us to reflect upon the data. This process of reflection contributed to the development of our results and provided us with support with regards to our theoretical sampling [44], for example, by informing our choice of participants' role that could provide more clarification about the concepts that we thus far generated.

We provide one sample from the category *Ability to systematise and transmit knowledge* to illustrate how we moved from raw data to the categories that we finally identified.

Open coding. In this first stage of the data analysis, we observed events, actions, and interactions in the course of the software development coordination activities [45]. We assigned these events, actions, and interactions with conceptual labels. After some refinement, they became the KBRs that were further grouped into categories.

For example, when we identified the concept *Conception of a knowledge retention structure* (see Figure 3.2), we observed re-occurring events in terms of the way that structuring knowledge impacted the reuse of code, an association between documents, and the logic that was used in describing parts of a system.

Concepts	Category
Identify what knowledge to keep in a systematic way	**Ability to systematise and transmit knowledge**
Experience transfer	
Recognise relevant knowledge	
Perceive what produced knowledge could assist other employees	
Balancing time allocation for systematise the knowledge and the time pressure for delivery	
Matching knowledge needs to the available knowledge	
Spreading the awareness of the existing knowledge	
Living document	
Representing knowledge efficiently into an artifact	
Knowledge retention structure	
Ability to integrate tool and coordination skills	
How architectural knowledge disseminates	

Figure 3.2: Example of the emergence of a concept

Axial coding. We identified additional concepts to *Conception of a knowledge retention structure* that also captured aspects related to storing knowledge and transmitting knowledge between individuals.

In the next stage of the analysis, as we reached theoretical saturation, we grouped the associated concepts into an abstract category *Ability to systematise and transmit relevant knowledge* (see Figure 3.3), and repeated the same process for the five remaining categories that emerged from the data. The remaining categories are: scenario analysis, social collaboration, task planning and resource management, team environment and settings, and inefficient utilisation of the knowledge resource. We gathered the concepts that fall under each category in table 3.2.

Phase 4 - Selective coding. After the third round of data collection, we began to delineate our core category by connecting it to the categories that we had created during the ongoing analysis. The core category was consolidated as soon as we reached theoretical saturation during the fourth round of interviews.

We diagrammed the categories and their relationships with each other by considering the elements for theory generation [45]. Figure 3.4 displays this

Sample of raw data	**Concepts**
"I think that for implementation, it sort of, store in the code . If you store in a structured way, it is reusable for next time . If next time we want to do something similar, we go back to that solution " *"we use different tools for support . We try to store knowledge in Jira, but sometimes we find the same thing in two different places "*	**Conception of a knowledge retention structure**
"we try to keep track of who is the best for a specific thing and where can we go if we need to do something fast " *"We didn 't have that many things written down because people knew each other for 20 years . I think that made us quite fast, but now we scaled up a bit, I begin to see stretches towards this culture because it is hard to have this many relationships "*	**Knowing what others know**

Figure 3.3: Example of the emergence of a category

process where the black lines indicate the link between the theory elements and the developed categories. This was done so as to explain a phenomenon that addresses the following: (i) the conditions that lead to the occurrence of the phenomenon; (ii) the circumstances within which the phenomenon occurs; (iii) the actions/strategy by which the phenomenon expresses, and the (iv) the consequences of the phenomenon. The core category emerged through a gradual and steady process of reflection. In this process, we discussed the connections among the categories, and we also made several adjustments to the theory.

In our research, the core category did not emerge from the existing categories. According to the originators of the method, this is quite a possible occurrence [45]. We observed that there is a predominant theme emerged during the data collection. Furthermore, this included the development of the categories that referred to constant changes to the product that motivated the continual need for the adaptation of processes and the application of new or existing knowledge. In this case, we needed a more abstract concept to explain the central phenomenon and to explicitly state the relationship between all of the categories that we identified [67]. This overarching concept was *Need for change.*

In the later stages of the selective coding, we consulted the literature to investigate the (potential) connections between the generated categories with

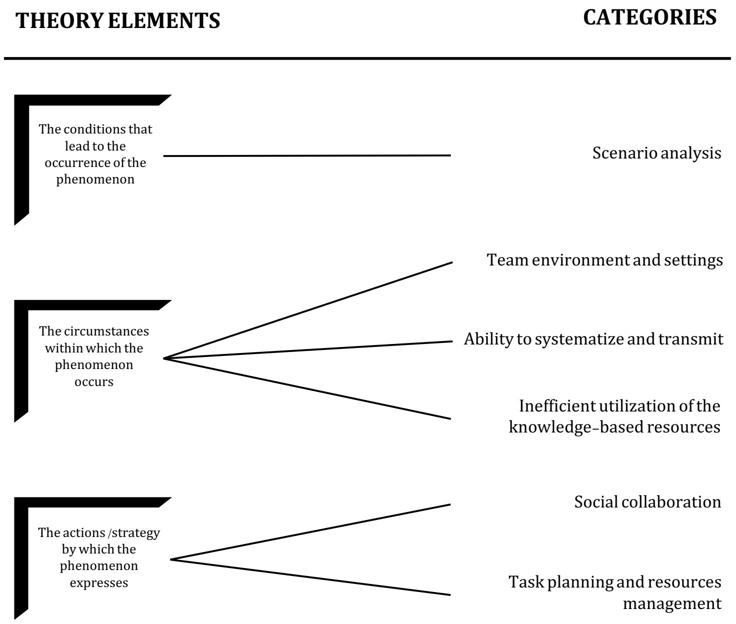

THEORY ELEMENTS
CATEGORIES
The conditions that lead to the occurrence of the phenomenon
Scenario analysis
The circumstances within which the phenomenon occurs
Team environment and settings
Ability to systematize and transmit
Inefficient utilization of the knowledge-based resources
The actions /strategy by which the phenomenon expresses
Social collaboration
Task planning and resources management

Figure 3.4: Illustration of the diagramming of the categories

previous research. In this process, we related our research findings to the existing literature to validate our theory (see subsection 3.3.3), to strengthen our understanding of each category, and to support our discussion (detailed in section 4.5).

Table 3.2: Skills that fall under each category

Causal conditions	**Actions/Strategy**	
Scenario Analysis	**Social collaboration**	**Task planning and resource management**
- ability to absorb changes that originate from the market	- Socialisation processes	- perspective of the product
- combination of technical capability with market vision	- sharing becomes priority	- product awareness
- knowing how the current technology should evolve	- personally characteristics	- comprehension of the implications of change during the software development
- balance between business and technical skills	- flat communication	- strategies to handle task planning that take dependencies into consideration
- evaluate the business value in the short term versus long term	- collaborative culture routine	- company's accumulated experience
- understand customer value	- cognitive processes for combining knowledge	- how to distribute human resources appropriately
- readiness to absorb changes	- level of control in activities that involve knowledge creation	
	- ability to conduct cognitive processes	
	- achieve particular goals when they are established	
	- joint effort for coordinating the transfer of technical knowledge	

Intervening conditions		
Team environment and settings	**Ability to systematise and transmit knowledge**	**Inefficient utilisation of knowledge-based resources**
- perspective of knowledge diversity	- identify what knowledge to keep in a systematic way	- meaningless search in a database
- combination of interpersonal skills	- experience transfer	- frustration because of recurrent problems
- coexist and interact with different personalities	- recognising relevant knowledge	- redesign solutions
- understanding of the cognitive processes	- perceive what produced knowledge could assist other employees	- knowledge loss
- management insight to apply suitable practices	- balancing time allocation to systematise the knowledge and the time pressure for delivery	- employee's knowledge gets attention mainly during a staff turnover
- team's learning awareness	- matching knowledge needs to the available knowledge	- disconnection from external environment
- knowledge's nature (tacit or explicit) directs the learning of the strategies that will be adopted by the teams	- spreading the awareness of existing knowledge	- unawareness of knowledge sources
- knowing what others know	- perception of a living document	- waste substantial time
	- representing knowledge efficiently in an artefact	
	- conception of a knowledge retention structure	
	- ability to integrate tool and coordination skills	
	- how architectural knowledge is disseminated	

3.3.3 Theory evaluation

Corbin and Strauss [45] established the criteria for evaluating qualitative research methods. They proposed two criteria classifications: one for the research process and the steps involved in this process (coding, memo, sampling), and another for the empirical basis of research findings. We addressed the criteria that are relevant to the research process by detailing each phase of the research design in the above [45]. We complemented this validation by following the guidelines for conducting Grounded Theory as proposed by Stol et al. [179].

We evaluated the empirical basis of our findings by means of seven criteria [45]:

- Criterion 1: *Are concepts generated?* Yes. We identified concepts that were grounded in the collected data.
- Criterion 2: *Are the concepts systematically related?* The way in which concepts are related to each other can be used to evaluate the robustness of the theory generated. We addressed this criterion through the narrative we present based on the data in each category. In turn, this shows how the concepts systematically relate to each other in the same category.
- Criterion 3 and Criterion 4: *3 - Are there many conceptual linkages, and are the categories well developed? Do they have conceptual density? 4 - Is there much variation built into the theory?* The categories are dense in terms of both the number of concepts and their relationships with each other. For example, the *Task planning and resource management* and *Social collaboration* categories act as action/strategy. The causal condition (*Scenario analysis*) and its concepts represent the events that lead to the occurrence of the phenomenon.
- Criterion 5: *Are the broader conditions that affect the phenomenon under study built into its explanation?* We took the broader conditions into account, specifically in the *Scenario analysis* category, which is the causal condition. This is explained by means of the concept of how the external environment affects the phenomenon.
- Criterion 6: *Has "process" been taken into account?* “Process” refers to the movement of action/interactions in response to prevailing conditions. We described how the condition (represented by one category) that gives rise to the phenomenon occurs through the action/strategy (represented by a pair of action/strategy categories).

- Criterion 7: *Do the theoretical findings seem significant, and to what extent?* The significance of the findings is described in terms of two main features: (i) the potential that the findings have to stimulate further studies and (ii) how the findings offer up a valuable explanation of the phenomenon under investigation. We gathered potential further research in Section 3.5.1. Furthermore, we consider that the theory that we present in this paper offers a valuable explanation of the phenomenon as a result of the application of a systematic process for conducting grounded theory.

To supplement the validation of the empirical findings presented in this paper, we mapped the categories that we identified to the existing literature. (The mapping is presented in more detail in Section 3.5.1).

3.4 Findings

In this section, we present and discuss the results of this study. First, we describe the theory that we created and each category that is treated by this theory in detail. In Section 4.5, we discuss our findings.

We gather our results in the **Knowledge-Push Theory**. The findings comprise the specific skills which represent the set of KBRs (detailed in Table 3.2) that are highlighted in italics and classified across five distinct categories except for the category Inefficient utilisation of knowledge-based resources, which represent complications that may arise from the mismanagement of knowledge-based resources:

- Scenario analysis
- Team environment and settings
- Ability to systematise and transmit knowledge
- Inefficient utilisation of knowledge-based resources
- Social collaboration
- Task planning and resource management

Table 3.3: The utilisation of the knowledge-based resources (RQ2)

Theory elements	**Categories**	**How do companies utilise knowledge-based resources?**
Causal conditions	Scenario analysis	To examine external variables that can affect the company, for example, a market fluctuation that demands adapting to respond to changes.
Intervening conditions	Team environment and settings	To understand how to set up agile teams that potentially create a favourable environment for knowledge sharing and collaboration - facilitating adapting to changes.
	Ability to systematise and transmit knowledge	To support identifying what knowledge a company should store in artefacts, its format and which tools should be utilised.
Actions/strategy	Social collaboration	To enhance social relations that can result in effective software teams and trusting environments to share and solve complex problems associated with software development.
	Task planning and resource management	To perceive the changes and their ramifications through the product development as its required adjustments to tasks and resources accordingly.

3.4.1 The Knowledge-Push Theory

Figure 3.5 depicts the theory that we developed in this study. The connections between the categories and the main phenomenon are clearly indicated. The categories are represented by identifiers (e.g., C1) which are discussed in more detail in the following subsections. The KBRs are distributed in ASD environments through five categories that comprise our knowledge-push theory (*RQ1*). The theory generation is informed by the explanatory elements of the phenomenon, as specified by Corbin and Strauss [45]:

- The *Phenomenon* is expressed by the Need for Change.
- *Causal conditions* the events which lead to the occurrence of the phenomenon that is expressed by the scenario analysis (C1).
- The *Context* is limited to software development companies that adopt agile practices or partially agile practices in their development activities.
- The *Intervening conditions* that explain the circumstances within which the Need for change occurs are represented by the categories: Team environment and setting (C2), The ability to systematise and transmit knowledge (C3), and The inefficient utilisation of knowledge-based resources (C4)
- The *Actions/strategy* by which the phenomenon is expressed are social collaboration (C6) and task planning and resource management (C5).

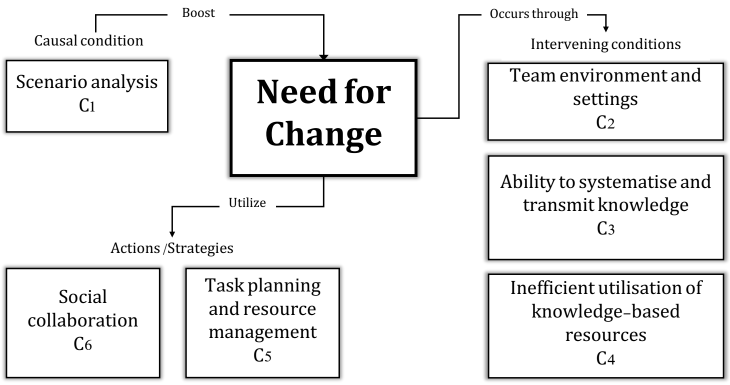

Boost
Causal condition
Scenario analysis C1
Need for Change
Occurs through
Intervening conditions
Team environment and settings C2
Ability to systematise and transmit knowledge C3
Inefficient utilisation of knowledge-based resources C4
Utilize
Actions /Strategies
Social collaboration C6
Task planning and resource management C5

Figure 3.5: The Knowledge-push Theory.

- The *Consequence* is the continuous assimilation of software changes through the application of KBRs.

From a high-level perspective of the phenomenon, our **Knowledge-push theory** reveals how KBRs boost the **Need for Change** in ASD. Changes that are made in the software product, for example, requirements changes that lead to feature changes, motivate the constant need to adapt previous planning and to apply new or existing knowledge by sharing, systematising, and transmitting knowledge to practitioners within and between teams.

The Scenario analysis (C1) determines, based on its KBRs, how a company responds to market changes and explores new opportunities. The primary strategies adopted by the practitioners are expressed through Task planning and resource management (C5) and Social collaboration (C6). The phenomenon occurs under the following circumstances - The intervening conditions by which these strategies occur are - Team environment and settings (C2), The ability to codify and transmit knowledge (C3), and The inefficient utilisation of knowledge resources (C4). Companies manage to use a certain level of their knowledge resources and consequently generate continuous assimilation of changes. However, poor codification and informal communication may result in significant knowledge waste.

The KBRs may be assigned different levels of importance. As we discussed earlier, even though two companies might have the same resources (both physical and intangible), the difference between them can be ascribed to their intangible resources and how each company deploys such resources [11]. Therefore, agile practices are dependent on KBRs to the extent to which how critical these resources are. How critical they actually are will vary based on the potential impact of the KBRs on the efficiency of coordination within the company. We summarised the utilisation of the KBRs (*RQ2*) in table 3.3.

Causal conditions

The causal conditions section describes the concepts that boost the Need for Change, which is the scenario analysis category in our Knowledge-push theory.

Scenario analysis - C1

The Scenario Analysis category displays practitioners' ability to logically examine external scenarios that might affect their current business model or might highlight new business opportunities. It boosts the need for change by expanding the analysis of diverse situations through the company's *ability to absorb changes that originate from the market* and the analysis of the impact of upcoming requirements.

Our study observed that competitive market conditions influence the agile practices that companies adopt and/or customise. Competing companies need to speed up their development processes and offer faster release cycles. This entails the adoption of faster development and shortened deadlines in response to customers' demands. One interview participant (P2), a program manager, stated that "the customer is starting to use it and starting to give us requirements, so these customers are putting heavy pressure on us. In the early days, it was easier; we had more freedom to develop at our own pace. Now the customer puts strain also on the agile way of working because as soon as you have a customer that signs a contract, he expects somethings to a certain date, and therefore, as soon you do that kind of agreement, you kind of destroy the whole agile flow."

Companies that are market leaders in terms of their innovative products or their ability to operate in less competitive markets have longer release cycles. Longer release cycles allow such companies to focus on product steadiness and long-term features. In this sense, the agile way of working is customised to cope with more flexibility with regard to innovation than dealing with the pressure to deliver.

The changes, in general, are primarily related to the continuous addition of new requirements. However, their selection requires a *combination of technical capability with market vision* if one is to predict the future of the products, as remarked by P9: "I came from the client-side. I used to program in one of these systems, and I think that knowledge helped me. I know how to view the product from the outside and how they want to utilise the product. What trends do we see?"

In addition, since the companies continuously implement new requirements, *knowing how the current technology should evolve* to support changes becomes a priority, as stated by P8: "That's one of our roles, to try to see where we are supposed to be in a few years on the technical support level."

Practitioners need to know how to *balance between business and technical skills* if they are to achieve a sufficient rationale for the decisions that they make. In this regard, P2 commented: "I have managed two programs until now, so I am quite new in the management kind of area, and it has been quite interesting for me to think about how my technical skills help me in this role." Practitioners can apply these skills to *evaluate the business value in the short term versus long term*, but also as they consider and *understand customer value* in the context of prioritising requirements, issues, and bug fixes.

Comprehending the implications of the changes that are made also influences the *readiness to absorb changes*, as pointed out by P5: "The collaborative work also applies to the layers, for example, managers. They all work with the same backlog and need to work together and make a decision on what they need to focus on, which committing to a sprint plan would not work well. They need the flexibility to diverge on the path they take due to maintenance and testing. They need to focus and fix before continuing towards the goal."

Intervening conditions

This subsection displays the three categories that are intervening conditions to the phenomenon explored in our theory: team environment and setting, ability to systematise and transmit knowledge, and inefficient utilisation of knowledge-based resources.

Team environment and settings - C2

This category refers to a software development team's environment with respect to the practitioner's knowledge and attitude. By gathering different perspectives together for problem-solving and performing tasks, this category provides a favourable environment for the phenomenon. Moreover, it facilitates locating knowledge sources.

The *perspective of knowledge diversity* contributes to the coexistence of different ideas. The goal is to entertain a variety of perspectives that can converge to give rise to new ones, resulting in new knowledge [139]. Regarding this, P8 commented: "I've been here quite a while, and there are practitioners that are here for two or three years, and I really want a group that is diverse in that sense and has different backgrounds in what they did.”

The convergence towards new knowledge within and across teams is moderated by the *combination of interpersonal skills* where practitioners not only communicate with each other but also *coexist and interact with different personalities*. There are two important aspects to take into consideration regarding this: the practitioner’s behaviour towards the *understanding of the cognitive processes* for combining diverse knowledge, and the *management insight to apply suitable practices* to stimulate socialisation. As remarked by P11: “If you are an extrovert person and you say a lot, it is important that you talk the right things and not just talk. We have more introverts, and that is alright. You need to know what their strengths are so that I can use them in the right context.”

A particular team’s environment can either facilitate or hinder the systematisation of unrevealed knowledge: a fact which enables companies to take ownership of the knowledge resource. In domains where skilled practitioners are required to perform an activity with a specific type of knowledge, the systematisation of said knowledge becomes even more critical. On this subject, P5 commented: “We will have a person leaving, he will be on parental leave for a very long time and probably. So, he will probably not coming back to our team again. He documents his ideas and thoughts on how to proceed and so on. We had another person leaving before who did a lot of security work. He sat down and documented a lot of that. We went through his patches and looked at that together to make sure we were not missing anything.”

The continuous assimilation of changes often relies on a *team’s learning awareness*, which positively impacts the expansion of the practitioners’ knowledge in agile teams. The *knowledge’s nature (tacit or explicit) directs the learning of the strategies that will be adopted by the teams.* The more complex the knowledge is in terms of externalisation, particularly tacit knowledge, the higher is the tendency to adopt more socialisation among the team’s participants. This may take the form of workshops or building communities for discussions. Prototyping is also a way of testing the combination of knowledge that has originated from cognitive processes and formal learning programs.

Finally, *knowing what others know* is a key element to consider when forming teams and collaborating within and between teams. When teams are collocated, knowing what others know is facilitated by the constant interaction between practitioners. However, in larger organisations, for example, with distributed development, knowing what each individual’s competencies are might be challenging but essential at the same time.

Ability to systematize and transmit knowledge - C3

This category relates to the practitioner's ability to recognise what knowledge (tacit knowledge and/or explicit knowledge) should be systematised into artefacts and how this should be achieved. This category also includes the ability to integrate coordination skills with tools. It offers conditions to the need for change by allowing teams and stakeholders to access relevant and structured knowledge when they need it without losing much time in the search process.

The different roles on different organisational levels recognise that to *identify what knowledge to keep in a systematic way* is highly relevant. However, it is acknowledged that it is challenging in agile contexts, where informal communication dominates. To illustrate this concept, we present two important quotes from practitioners: I) P8 reported: "There is always a problem when you have new people or when you make something completely new that doesn't fit your way of working. That isn't easy to map to the common knowledge. We started to see that there might be a problem now that we are expanding geographically. We see that we need to change, but we need to know what is important to be in formal documentation and process." II) P3: "There is this thinking, not only in this company, that I write my code and I did my job. However, in a large organisation, it is difficult to maintain a product without documentation. One team develops the code, and another team tests it. Then a different team from the first one needs to investigate it two or three times more to start to fix the problem because they need first to understand what the first team has done, due to the bad documentation."

To a large extent, knowledge is shared, created, and applied in daily routines. Flexibility in communication, together with a collaborative environment, promotes the *experience transfer* by combining practitioners' experience and backgrounds. Quoting P11: "[...] in fixing the teams, I am looking at ages, experienced people. I try to hire new people to learn from the older people." The experience transfer promotes learning and knowledge creation by combining different expertise and experiences.

However, practitioners frequently struggle with *recognising relevant knowledge* for two main reasons: First, difficulties may be encountered as they *perceive what produced knowledge could assist other employees.* Second, practitioners may find it challenging *balancing time allocation to systematise the knowledge and the time pressure for delivery.* Regarding this issue, respondent P1 stated: "We really haven't that culture at all, we have tried to document stuff, but we are so decentralised that if we introduce something it has to cause less friction, and if there is no immediate benefit on it, it would not fly."

To ensure that practitioners within and between teams do not waste time looking for existing knowledge, one should establish procedures for *matching knowledge needs to the available knowledge.* However, we notice that this is still an open issue in the companies that we interviewed, together with the challenge of *spreading the awareness of existing knowledge.*

Further to the above, systematised knowledge calls for the *perception of a living document* to keep knowledge updated, as indicated by P3: "Each team will document a part of the code so when new people come they know what that part is about, and have the idea that you do the documentation and that's it. It is a living document" Nevertheless, retaining this knowledge involves *representing knowledge efficiently in an artefact.* This is facilitated by a *conception of a knowledge retention structure*, so it can be reused and represented in an uncomplicated way, as remarked on by P14: "Everything is relevant in some way, but it has to be in a structured way."

Although there are several tools available for managing software development knowledge and the accomplishment of tasks, the *ability to integrate tool and coordination skills* works as a backdrop for breaking down backlog items and disposing of them coherently within the tools, as P1 clarified: "The product owner, he is very detailed, very structured, so it is very easy to know what they are doing, they have a very good planning in the 'project management tool'. It is very visible, and they are also very good at understanding the big picture and the value they would be adding by solving this."

In addition, this ability can also be associated with *how architectural knowledge is disseminated.* Practitioners rely on the overall design systematized in a formal language and distributed in the company. However, it might be misinterpreted by teams and cause mistakes, as pointed out by P18: "It is quite easy to misinterpret the architecture in software development. The architects define the architecture, and they rely on the team to do what is designed."

Inefficient utilization of knowledge-based resources - C4

Software companies achieve continuous assimilation of changes by using agile coordination mechanisms. However, the use of knowledge-based resources often remains an inefficient process. Even though there is an existing awareness of the relevance of knowledge systematisation, practitioners frequently face difficulties in recognising what knowledge to codify into an artefact in daily activities and when they should do so.

In combination with these challenges, practitioners also sometimes codify irrelevant knowledge. This practice occurs mainly through software and databases, which are the most frequently used technologies for codification in software companies. The result of this codification of irrelevant knowledge is a *meaningless search in a database* which is not updated and is not entirely trusted by its users. In this regard, P16 reported that: "[m]ost of the time, you get 2000 results and go through, you search. The first 200 items are outdated, and below that is misinterpreted, so not really the right one."

Being committed to fast delivery through iterations may bring about increased inefficiencies in terms of KM in coordination activities, giving rise to *frustration because of recurrent problems.* Note that this category refers to problems that were entirely or partially solved previously. The practitioners whom we interviewed mentioned that, in several cases, they would *redesign solutions.* For example, note what P17 remarked on this: "Of course, it happens[...]you can search what others have done in the past, but we have a database that there are thousands of issues. How do you search in the database?"

One important factor that frequently results in *knowledge loss* is the fact that an *employee's knowledge gets attention mainly during a staff turnover.* In this case, employees with specific technical knowledge, for example, in the domains of security or streaming, occasionally are requested to codify "what they know" in relation to a particular topic in a short time. In this process, knowledge is partially lost, thereby affecting the other employees' learning time.

Inefficiency in the management of the knowledge resource also affects the competitive positioning of a software company in a market. In particular, when software companies are market leaders or closely compete with the leaders, the inefficient coordination of requirements engineering activities might motivate the company's *disconnection from external environment.* This scenario is usually characterised by an *unawareness of knowledge sources* for eliciting new requirements through cognitive processes. Note the remark made by P9: "Sometimes it is very hard to see what business value this brings, short term versus long term. There are one or two layers between the functionality I add to the actual business and selling, so it is quite hard to know. I go with my gut feeling a lot."

Finally, using the different coordination mechanisms and activities within ASD, companies *waste substantial time* searching for relevant knowledge that might be already codified or known by their employees. In this regard, P3 commented: "The documentation is poor, each team keeps their repository, and things get worse when you go to other departments."

Rus and Lindvall [161] argue that KM could prevent and mitigate risks in software development companies related to knowledge loss, a lack of knowledge, performing re-work and solving problems that have already been solved, and staff turnover.

Inefficiencies related to the management of knowledge resources have been examined previously in the literature [51,64,90,168], but our findings also reveal additional inefficiencies, including the redesign of solutions due to an unawareness of knowledge sources. In Section 4.5, we expand on this discussion by providing a number of potential implications that stem from these inefficiencies.

3.4.2 Actions/strategy

This subsection describes the categories by which the Need for Change is expressed: task planning and resource management, and social collaboration.

Task planning and Resource Management - C5

This category refers to the employees' understanding of (i) change, (ii) the ramification that their actions may have throughout the task planning process, and (iii) available resources. The Need for Change expresses through this category by practitioners' coordination actions, taking the product's complexity in its surrounding ecosystem into account. The *perspective of the product* is a knowledge-based resource that allows teams to reduce waste. In this context, waste refers to unnecessary code and time spent on fixing problems introduced to the software development project by losing the perspective of the product.

Lacking *product awareness* is a phenomenon that several practitioners face, as reported by P6: "In this project, it is common when focusing on specific features, the developers lose the idea of the product and end up developing unnecessary code that will cost more time on refactoring. People do not stop to analyse what is being done."

When new requirements are incorporated into the product, practitioners must possess *comprehension of the implications of change during the software development* if they are to verify how a new requirement (change) impacts the development, testing, and deployment time of the software product. As P1 remarked: "There are technical aspects for features that we introduce that I join to study what would be changed to the product, what would be adding to the product, how would it impact the deployment and of course the tests. It is quite a lot, actually."

To increase their understanding of the product, companies establish *strategies to handle task planning that take dependencies into consideration.* We observed that the companies included in this study made changes to their agile practices when they noted the appearance of several dependencies. These included dependencies related to product growth and distribution with other teams or sites. P13 observed: “When the project was mainly concerning us, it worked quite well. But as soon we got dependencies, and project managers had different priorities, it didn’t work quite well. That’s one of the reasons why we switched to Kanban.”

One strategy that can be implemented to deal with dependencies is to plan the tasks that need to be completed through iterations and prioritise tasks in each iteration. In this regard, P2 commented: “There are maybe five departments, and all of them need to do a piece of work for this to be complete. And, therefore, this department needs to go first, then this one, these two can go in parallel, and then I can do my work. So, there is a lot of planning to this to be done in that way — dependency planning. So, this goes through several iterations, I can say. That’s how we plan our work.”

Systemic reasoning with regards to software products is reinforced by the *company’s accumulated experience.* Experienced practitioners can support coordination (as described by P2 above, for example) by possessing broad knowledge about the product. These experienced engineers often take on decision-intensive roles, for example, by being responsible for lead architecture, project management, and software quality.

Establishing a team requires expertise from managers with respect to *how to distribute human resources appropriately.* Managers should allocate practitioners to a team to fit the teams’ purpose but also take into account their knowledge of the product they are tasked to produce. With respect to the allocation of human resources, P2 commented: “It is actually a combination of all the departments. Therefore, you need some teams that we put together that have connections with all other departments and have forced departments to act because sometimes these departments might think this is not our problem, this is their problem. Our performance is fine [referring to the practitioner’s team’s performance], yes, but if you put them together [all of the teams], the performance is poor.”

Social collaboration - C6

This category refers to practitioners’ social skills as they engage in information and knowledge exchange and networking activities. This category manifests the need for change by providing sociable and trusting environments where practitioners experience effective guidance for their working routines.

Socialisation processes play an essential role within a software development team by shaping the practitioners' behaviour in a way that promotes improvements in their communication. Social collaboration does not entail that introverts are converted into extroverts. Instead, it addresses how one can manage social processes where *sharing becomes priority* taking into consideration the practitioners' *personally characteristics.*

Companies that adopt ASD benefit from *flat communication*, which enhances a *collaborative culture routine.* At the same time, the hierarchical structure found in different departments and roles does not undermine the potential for communication among practitioners. As stated by P14: "You can talk to managers and managers of managers like normal people discussing things, you can discuss with people on other levels."

Flexibility of communication within and between teams triggers *cognitive processes for combining knowledge* from different people and roles. This knowledge can be used to deal with the implications of change. Regarding this, P14 commented: "When discussing something, maybe two people who do not know the problem, when they discuss things together, then they solve the problem." In addition, the awareness of the cognitive process for combining knowledge is associated with the *level of control in activities that involve knowledge creation.* At higher levels, this may hinder creativity, as noted by P16: "It will stop creativity if you are too formal, it should be a bit loose, but not too loose because it will not gain anything good."

Meetings can also trigger cognitive processes in an agile community. The *ability to conduct cognitive processes* may drive people to *achieve particular goals when they are established*, for example, by combining different perspectives when decisions are made. In this regard, P2 commented: "My personal hate is re-occurring meetings. If I had a choice, I would cancel all re-occurring meetings and forbid them because 90% are a waste of time and could be handled in a better way through direct communication or whatever. Of course, there are meetings that when issues arise, and it is simply the best way to put everybody in the meeting room, bash their heads together, and the issue is solved."

When systems from different companies need to be integrated with each other (from partners or a customer), a *joint effort for coordinating the transfer of technical knowledge* goes beyond merely breaking down the work to be done. In this scenario, collaboration aims at solving issues together through cognitive processes that stimulate knowledge creation and its application.

3.5 Discussion

This section discusses in subsection 3.5.1 how our findings relate to and add to the existing literature on KBRs and software engineering. In subsection 3.5.2, we also provide a discussion on how to consider knowledge as a resource in an industrial context, connecting the results of our study to a collection of practices that can optimise the utilisation of this resource.

3.5.1 Connections with existing literature and implications for research

Our work expands on previous research on KBRs by providing an increased understanding of the identification and role of KBRs in ASD. Our findings also confirm the results of several previous studies [97, 134, 138].

In periods of stability, companies benefit from property-based resources. However, in changing and unpredictable environments, KBRs have contributed to increased profits and sales [134]. The category *scenario analysis* (C1) highlights the importance of KBRs in dealing with a company's external variables. Our study suggests that KBRs contribute to improved coordination when they are used to make adaptations to current processes and to grasp new opportunities in response to market changes. In the software engineering literature, we identified several different aspects of external scenarios that affect the initial stages of software development activities. These aspects include: software product management [101, 124], requirements engineering [48, 77, 95], and market-driven software development [72]. However, although these areas relate closely to our category, the existing literature on software engineering does not explore how practitioners can benefit from KBRs in a changing environment such as ASD. Further studies exploring the initial stages of software development could consider these findings. For example, market-driven companies [72] could incorporate their practitioners' technical capability with regards to their market vision skills to the requirements elicitation phase and then measure the improvements that derive from such incorporation of technical ability.

Previous research has shown that human interaction intensifies product innovation and organisational innovation through the generation of new ideas. Efficiency can also be improved by sharing knowledge, and skills [138]. Our findings, more specifically the categories *social collaboration* (C6) and *team environment and settings* (C2) relate to previous research in this area. ASD team structure influences creating a network environment, where social relations are a primary channel along which tacit knowledge can be shared. Good social relations can also be used to predict the effectiveness of software teams [18, 26, 148, 161, 162]. Good social relations rely on heterogeneity in terms of knowledge diversity and interpersonal skills to effectively solve complex problems, which is undoubtedly part of the software development process [26]. In this context, we should mention that behavioural software engineering is a recent but growing field. The areas that are explored in this field are closely related to the concepts that we have described in this category, including group thinking and team composition [115, 116, 172].

Encouraging collective thinking can be viewed as an alternative to triggering the cognitive processes for combining knowledge [141]. As people share and consolidate knowledge, part of this knowledge remains tacit, whilst part becomes a property-based resource through codification practices. Agile teams utilise several Information and Communication Technologies (ICTs) for this purpose. However, it is not made clear in the literature how knowledge is effectively employed in the codification processes [36,57,94,105,148]. In the category *ability to systematise and transmit knowledge* (C3), we explore this gap by identifying several KBRs that can be used to support coordination activities. They are represented by skills that are related to understanding what knowledge one should store and also to understanding the extent to which knowledge should be codified, be it entirely physical, digital, or a mix of both media [34,49,55,62,173]. Future empirical investigations in this domain should focus on how a balance between socialisation and codification practices can be taken into account. It should also be possible to verify how the complexity of increased dependency between geographically distributed teams affects this balance. Finally, it is also relevant to know how this complexity affects the cost of codification practices.

Although the category of *task planning and resource management* (C5) is a new contribution to the literature on KBRs, it has been subject to no small amount of attention in the general software engineering literature. The absence of a holistic perspective of the product and the associated business model is a common issue that ASD faces due to extensive focus on delivering features rapidly at the end of the development sprint and the ubiquitous presence of time pressure [15, 29, 93, 121]. Borrego et al. [25] argue that poor design documentation results in the loss of architectural knowledge, which they define as knowledge vaporization. Our findings offer additional clarification concerning how KBRs support a broader understanding and diffusion of the product. Diffusion can be achieved in different ways, for example, by formal visualisation techniques in ICTs could be more effective in distributed teams since this would increase the probability of reaching a larger number of people, while on the other hand, informal interpretations that developers make in co-located teams could cost less and also could be effective [153]. Because KBRs might affect a company's performance [16], further empirical work is required to examine the efficiency of KBRs that are aimed at verifying how much knowledge has been wasted or is absent in crucial moments.

3.5.2 Practical implications for KM in ASD

Although the concept of 'knowledge' has been discussed in several software engineering studies, there remains only a weak connection between the management of this resource and coordinating activities in ASD. At present, there is a lack of research that is focused on how to apply knowledge as a resource [148].

From the strategic management point of view, the main implication of the above is the impossibility of assessing KM effectiveness. For example, 'product awareness' (See 3.4.2) is a valuable knowledge resource. Its mismanagement may have implications with respect to writing unnecessary code. Most often, ASD focuses on this aspect informally and relies on the individual employee's perception of an event. Note that different employees may well not share the same insights.

An alternative to verifying the extent knowledge contributes to generating business value by addressing mismanagement issues is to incorporate KM practices into ASD coordination activities. This will enable one to develop strategies that can be used to analyse the application of resources [146].

Knowledge management practices should be developed with the aim of achieving a purpose. Furthermore, they need organisational support and stimuli [164]. Leadership plays a fundamental role in stimulating and creating an environment where knowledge can be efficaciously managed.

Similar to property-based resources, knowledge also needs to be managed so as to ensure that its application is effective. In this respect, an informed leadership team can guide companies to actively and dynamically apply knowledge in their daily work by creating the appropriate conditions [141].

Although ASD employs somewhat flat hierarchical structures and emphasises shared responsibility within the teams, distinct leadership roles exist in different methods and frameworks (for example, product owners and scrum masters). Besides agile-specific roles, typical roles found in software companies are also qualified roles, including line managers, project managers, and architects.

The presence of middle managers and alternative leadership roles can enable the synchronisation of knowledge goals within an entire company [141]. Proper leadership can mediate processes at each company level by aiming to enhance knowledge creation, storage, sharing/transfer, and application [148].

We suggest two complementary strategies with respect to the management of knowledge in ASD contexts, namely (i) the mapping of knowledge sources and (ii) knowledge codification. We summarise these strategies in Figure. 3.6.

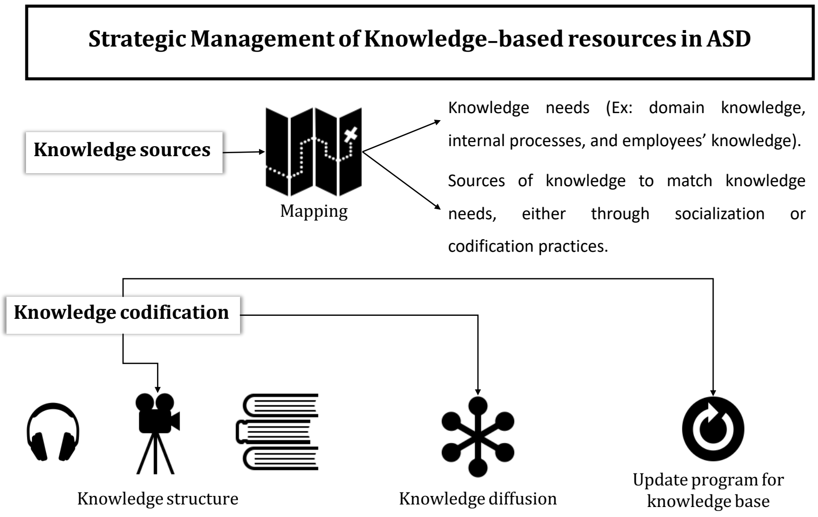

Strategic Management of Knowledge-based resources in ASD
Knowledge sources
Mapping
Knowledge needs (Ex: domain knowledge, internal processes, and employees' knowledge).
Sources of knowledge to match knowledge needs, either through socialization or codification practices.
Knowledge codification
Knowledge structure
Knowledge diffusion
Update program for knowledge base

Figure 3.6: Recommendations for managing knowledge

Knowledge codification

The codification of knowledge in ASD involves different levels of formality, which are guided by the company's documentation strategies and policies. Note that most documentation currently focuses on documenting feature specifications, tests, changes, problems, and solutions.

Besides the company's codification policy, most teams make a record of what they believe is relevant in repositories. Formal artefacts are commonly recorded in software that the company provides. Artefacts that are adopted informally are selected by the team themselves, for example, in the form of Wikis. However, a lack of precise knowledge structure and guidance may result in wasted time and a resultant economic waste. Several practitioners (P1, P3, P9, P14, P16) agreed on the importance of recording knowledge in some form or another to attend to the needs of different types of users and applications. Notwithstanding this, we currently lack the expertise in how this may be best achieved.

Developers in software development teams perceive the codification process as a distraction. Stettina et al. [178] found that codification activities were given to the least qualified developer, while the rest of the team focused on what they thought was important, i.e., writing code. The time pressure impacts people's behaviour, forcing them to focus on very specific features. Consequently, they often pay less attention to seemingly peripheral—but not less important—tasks. The result of this behavior can contribute to documentation debt [8, 59, 187].

In contrast, one study has indicated that decision processes that are related to the cost estimation of future projects may be based on a series of miscalculations due to a lack of codified knowledge. Apparently, this is predominantly the case in the requirements engineering phase [163].

Whilst knowledge codification can be costly when appropriately planned and implemented, it can positively affect innovation, economic growth, and knowledge creation [10, 41].

Cohendet and Steinmueller [41] explain that the use of information and communication technologies (ICT) significantly reduces the cost of knowledge codification and also facilitates the dissemination of this knowledge. Additionally, information and communication technologies are of use in transforming knowledge application into routines.

Agile teams who employ ICTs to codify and share knowledge should pay attention to the following three points:

- *Match codified knowledge to people's knowledge needs.* Teams work on different items at different development phases, which is characteristic of a complex environment where several knowledge sources and several potential users are present. Thus, the point of matching codified knowledge to people's knowledge needs includes the contexts in which the knowledge should be accessible, whom. Furthermore, the design and the usage of ICTs should be viewed as a source of guidance to relevant job-related knowledge [82].

- *Knowledge structure.* This point relates to the form of the knowledge when it is codified into an artefact and how it affects the usability and applicability of the knowledge by the user. According to Hall [76], two main aspects are critical to knowing how to codify knowledge. These are (i) understanding why people need certain knowledge and (ii) knowing how it might be applied.

- *Knowledge update.* To ensure that the knowledge matches the intended user's needs, one must regularly update previously codified knowledge

[106]. The assimilation and accommodation of change is an ongoing process in ASD and thus creates different knowledge needs.

It is important to note that one should not assume that *codification* is a synonym for *ICT*. The existence of knowledge that has been codified into artefacts does not guarantee its applicability. Whether knowledge is applied or not depends on a range of social and cognitive aspects.

Notwithstanding the above, it is likely that an appropriate codification process that is aligned with the effective use of the knowledge [3] can result in reduced time wasted and reduced deployment delays by speeding up the knowledge retrieval process.

Knowledge Sources

Knowledge has different levels, sources, and associated goals. For example, these may include high-level goals and strategic directions, specific requirements, or a specific source code [161].

From the KM perspective, most ASD companies fail to adopt practices that specifically aimed at identifying knowledge needs or at satisfying particular knowledge needs by using knowledge sources. As discussed in the previous section (see 3.5.2), most knowledge codification activities are guided by company policies that result in minimal product documentation that is often outdated.

Except for the knowledge that is codified in the artefacts, the identification of knowledge needs in ASD contexts is usually made via informal communication between people either inside the software development team or between teams. Note that most developers tend to communicate with a limited number of people whom they already know. However, they might not find the sources that they need to acquire knowledge.

From this inconsistency between the knowledge needs and knowledge sources, two scenarios emerge: (i) practitioners have confidence in the knowledge that is stored in artefacts and (ii) a lack of awareness of the knowledge that is possessed by others.

Several practitioners who were interviewed for this study (P14, P17, P18) mentioned they prefer not to consult their databases as a knowledge source in their search for similar problems/solutions because they are too large and outdated. This happens because a recipient's behaviour towards a knowledge source is directly influenced by how reliable the source is [6, 184].

Reliability (or perceived reliability) also affects the degree of cooperation that takes place between people as sources of knowledge. The more reliable the source is considered to be, the more cooperation takes place. This is due to the notion of ‘receptiveness’ [56]. A high degree of receptiveness can lead to reduced knowledge exchange costs [47].

A second implication relates to a lack of awareness of the knowledge that other people possess. In small companies with co-located teams, the competencies of the team members are usually well known. In such contexts, it is easier to find knowledge sources that meet knowledge needs. However, in contexts where the number of teams and employees do not allow this type of close connection between people, the time that is spent on solving problems might increase, and, in extreme cases, the cost may supersede that of the cost of acquisition of external sources of knowledge.

In all of the companies included in this study, despite the fact that they employ line managers or similar, team members are often unaware of their colleagues’ knowledge or competencies with respect to their day-to-day work. Another point made by P2 during the data collection phase of this study is that by knowing what the others know, one is more likely to more precisely plan resources since one is more likely to be aware of the “workload capacity” of each individual in the team.

This analysis is supported by Rus and Lindvall [161], who identify five knowledge areas in software engineering that are critical to achieving business goals. These knowledge areas are: (i) knowledge of new technologies, (ii) domain knowledge, (iii) knowledge of internal practices and policies, (iv) knowing who knows what, and (v) collaboration for sharing knowledge.

Knowledge mapping is an essential step in managing knowledge sources [171]. This includes identifying relevant sources of knowledge that can be used to bridge knowledge gaps between people at a company. This mapping can be executed by external knowledge sources, such as new technologies and market trends, and then deployed internally to identify who knows what and where knowledge is missing.

3.6 Threats to validity

In this section, we discuss a number of threats that the validity of our study is faced with. In this context, we have followed the guidelines recommended by Wohlin et al. [200], who classify such threats as being external, internal, conclusion, or construct threats.

External validity relates to the possibility of generalising the results of the study in settings that lie outside the original study’s settings. Even though we interviewed 18 practitioners from five different companies, we are aware that this sample is not representative of the entire software industry. Notwithstanding this, we aim to achieve a certain level of analytical generalisation [66] by providing rich contextual information and a close discussion of our findings.

Our strategy to further mitigate external validity threats was to contact companies that develop software that is combined with hardware and companies that produce software only as their main product. In addition, we also selected companies that operate in different domains. We believe that this strategy of combining domains, product type, and market conditions supports the generalisation of the results of our study.

The **internal validity** of this study relates to our awareness of other factors that might affect the casual relationship that we have investigated. Our study is exploratory rather than confirmatory, which minimises some internal validity threats.

We recognise that the fact that each step of the Grounded Theory relies on the researcher's subjectivity is a validity threat. To diminish this threat, we followed the systematic process described by the Grounded Theory [45]. The design phase of this study was an interactive process. The authors participated in commenting on and adjusting the study's research proposal and the data collection instrument. In the data collection phase, the first author conducted the interviews that were further discussed with the second author (see Section 3.3).

The **conclusion validity** threats to this study are related to factors that might interfere with our drawing accurate conclusions. In our study, we recognise that a potential lack of consistency in the concepts during the data collection was a threat. To mitigate this threat, we decided to collect the data in batches. By adopting this strategy, we executed the coding after each round of the data collection phase and compared our coding to the concepts that we had identified previously. This approach allowed us to achieve a certain degree of consistency with respect to the concepts that we identified throughout the analysis (see Section 3.3).

Although Grounded Theory provides ways for validating theory generation (see Section 3.3), we provided each category with a synthesis discussion, including how the existing literature has treated the concepts that originated from our analysis. In the synthesis section, we discuss similarities and contrasts in software engineering and other areas of the relevant literature.

With regards to **construct validity**, we discussed possible threats that may lie between the research setting and the theoretical construct that we explore in this paper. One threat that we identified was the inadequate pre-operational explication of the constructs from different areas, including 'knowledge management'. To reduce this threat, during the interviews, we refrained from using the specific terminology related to KM since such terminology was not common knowledge for the practitioners (see Supplementary Material A). We applied the terminology in our analysis, of course. For example, when we use *knowledge diversity perspectives*, we refer to people with different backgrounds inside the team.

To avoid mono-operation [200] bias, we interviewed a wide range of people who held different roles and worked with different agile methods. Concerning evaluation apprehension, before the interviews started, we explained to the practitioners that they would be anonymised and that the results of the interviews would be combined so that it would not be possible to associate any information with any one particular practitioner. In addition, we explained to the practitioners that, since the study was an exploratory study, our goal was not to evaluate their knowledge management but, instead, our goal was to come to some understanding of how the knowledge management took place.

3.7 Conclusions

Knowledge is recognised as a significant resource for software development. However, a lack of understanding of how one should manage knowledge can hinder its effective use in a software development context. We examined knowledge from a resource-based perspective, which gave us insight into how this intangible resource is relevant to ASD contexts.

By following the systematic process outlined in Grounded Theory, we identified a number of KBRs in their different forms, including specific skills. These KBRs were then gathered together into the Knowledge-Push Theory. This theory provides an explanation of how practitioners use the identified KBRs to boost the need for change in ASD. The explanatory potential of the theory was validated through a comparison between the categories that we established and the existing literature on this topic.

The results of our study show that, as primary strategies, practitioners use task planning, resource management, and social collaboration. These strategies are implemented through the team environment and settings and are made manifest in the practitioners' ability to codify and transmit knowledge. However, this process is non-systematic, which brings inefficiency into the domain of knowledge resource utilisation, resulting in potential knowledge waste. This process can generate negative implications to the course of software development, including meaningless searches in databases, frustration because of recurrent problems, the unnecessary redesign of solutions, and a lack of awareness of knowledge sources.

To employ the theory presented here, practitioners must note that, similar to any other type of resource, knowledge-based resources enjoy different levels of importance for each software company. As a starting point, we suggest that practitioners prioritise *critical* KBRs and develop strategies to manage these. The strategies that are ultimately selected could be thought of in terms of the main implications that we describe in Subsection 3.5.2: namely, knowledge codification and knowledge sources.

Regarding future research, we suggest that two main issues be explored. The first relates to the definition of metrics that can be used to evaluate the outcomes in both the codification and personalisation of KM strategies. The second issue is related to finding a balance between (A) informal rules for communication and (B) strict rules for communication and codification and how such a balance may affect the agile flow?

Acknowledgment

We would like to acknowledge that this work was supported by the KKS foundation through the S.E.R.T. Research Profile project at Blekinge Institute of Technology.

Chapter 4

An Investigation of Causes and Effects of Trust in Boundary Artefacts

This chapter is based on the following paper:

> Raquel Ouriques, Fabian Fagerholm, Daniel Mendez, and Baldvin G. Bern. "An investigation of causes and effects of trust in Boundary Artefacts". *Information and Software Technology Journal*, 158, 107170, (2023).

4.1 Introduction

Boundary Artefacts (BAs) are central objects that tie together teams in multiple social worlds along the software development lifecycle while holding different meanings [27, 165]. They are used to steer collaborative work between various stakeholders. For instance, a use case describing how users intend to interact with the system is defined from a requirements perspective. It serves as a basis for various other activities such as project organisation and management (e.g. effort estimation), design, and (acceptance) testing.

The BAs are essential for effective collaboration because they contain relevant information that supplies different groups with different needs. They materialise mainly as electronic or printed documents and, for the most part, are produced and used by humans. Being central resources and extensively used, disregarding the methodology adopted to guide software development processes, they provide value by condensing practitioners' knowledge in different formats, such as architecture descriptions, requirements specifications, test cases, etc [147, 201].

Software development teams, especially when geographically dispersed, rely on BAs for efficient coordination and knowledge sharing. The successful handling of such artefacts depends on how people manipulate and interpret information, and how they use technology to spread knowledge. Such complexity increases when a BA requires contributions from different social groups with distinct perspectives and knowledge background [2, 202].

While providing benefits such as shared understanding and knowledge resources to distinct groups, BAs can also fail in their purpose. Misunderstandings can happen when the artefact is not plastic enough to accommodate the different meanings. In other cases, it requires more contextual information so stakeholders can process the available information, or constant updates are necessary to satisfy needs [13, 81, 111].

As such inconsistencies occur, stakeholders' usage can be undermined, and the efficiency of these artefacts is reduced. For example, when a BA is mismanaged, learning across diverse groups becomes limited and integrating additional knowledge is difficult [118]. Moreover, when a BA do not comply with sufficient level of detail and practitioners neglect the management of such artefacts, stakeholders won't be able to find relevant and updated content. In worst cases, the workload can increase due to redesigning solutions that could be avoided if artefacts were properly managed [147].

In this circumstance, when practitioners do not feel confident in using the BA, trust decreases, and unknown consequences to the software projects can come into sight. This situation can also create an unpleasant environment with a weak sense of community that can be developed and encourage demand for accountability [2, 20, 190].

As trustworthy BAs are crucial for executing software development activities successfully, it is important to understand how practitioners create and use them, and what the implications are when there is a lack of trust in the BAs. *Trust* is an essential factor that determines the adoption of artefacts and the extent to which a practitioner feels confident using it [108, 190].

Software engineering researchers have examined BAs in several past studies. Most studies explore how companies benefit from using them for information sharing and collaboration among teams [74, 86, 149, 196]. On the other hand, BAs must be trustworthy so that practitioners can rely on them to execute their tasks. However, to the best of our knowledge, this aspect has not been examined so far.

In this investigation, we address this gap by reporting on a study examining the creation and utilisation of one specific BA and analysing how trust affects stakeholder behaviour towards it.

In particular, we aim at the following contributions:

- An empirical investigation of how a BA is created, utilised, and managed in a software development environment.
- An analysis of the implications of a decreased level of trust and how they influence the stakeholders' behaviour towards the BA.
- A discussion of the possible implications for future research and potential solutions for managing the analysed BA.

This manuscript is organised as follows: In Section 6.2, we present a brief background to our study and discuss related work. Section 4.3 describes the research method. In Section 4.4, we present the findings of our study. In Section 4.5, we discuss the implications that our findings have and future research. In Section 6.7, we then discuss the threats to the validity of our study before concluding our manuscript in Section 4.7.

4.2 Background and Related Work

In this section, we first introduce the basic notion of boundary artefacts (BAs) as background of our study before discussing related work.

4.2.1 Boundary Artefacts

The concept of boundary artefacts is not new and originates in the discipline of ecology [111, 174, 175]. Two main terms are commonly used in the literature: Boundary objects and Boundary artefacts. As we focus on the software engineering discipline, we use the term *boundary artefact* (BA) due to its proximity to the field. While we refer to Mendez et al. [132] for a more elaborate discussion and definition of the term, an artefact is, in simple terms, a work product that is produced, modified, or used by a sequence of tasks that have value to a role, such as code, test cases, or a requirements documentation.

The original introduction of the term boundary artefact roots back to 1989 and was coined in a study developed for the Museum of Vertebrate Zoology

[175]. The authors define it as "objects which are both plastic enough to adapt to local needs and the constraints of the several parties employing them, yet robust enough to maintain a common identity across sites".

The boundary does not refer to a limit, edge, or periphery but a shared space structure. These objects carry an interpretative flexibility characteristic, which means that different groups gather information and interpret it differently from the same object [114].

Star and Griesemer [175] detail four types of BAs:

- Repositories, which are traditional and standardised piles of objects such as a library.
- Ideal type, which denotes objects with a high level of abstraction that are adaptable to all domains. However, due to its vague characteristic, it suits better for communicating.
- Coincident boundaries, i.e. objects which, while possessing different contents, have the same boundaries. They fit organisations with geographically distributed stakeholders that need the information to work autonomously while cooperating with a commonly shared referent.
- Standardised forms, which are are objects with a rigid structure for adding information.

BAs are, thus, useful objects that can support software development in many aspects, including being the source of information to many stakeholders within an organisation. However, the fast-paced changes to the software products can affect the flow of information in these objects [181].

As boundary artefacts grow, they can lose flexibility because too many boundaries are crossed. When there is an overload of meanings, a BA can lose value to its stakeholders, resulting in workarounds to avoid the BA at all [81, 175, 206].

Practitioners need to manage the different borders and information needs of other stakeholders to keep the BA useful with shared understanding [20, 202]. As we recognise the importance of BA for software engineering, we want to understand why people avoid using them. The circumstances around the creation and management of the BA can affect how people trust them, and as a consequence, the BA can be more or less valuable or attractive, which may explain why they are used or not.

4.2.2 Work Related to Trust in (Boundary) Artefacts

As the notion of trust has been explored in several disciplines, there is still no consensus about its exact definition. The term acquires different meanings depending on the context investigated [20, 108].

A large number of definitions of trust connect to human behaviour towards another. In this context, trust refers to how individuals depend voluntarily on other individuals' behaviour. Due to its vague trait, it is complex to evaluate. To make it possible to describe, several authors proposed *trusting beliefs*, which are favourable factors that help to describe what causes a person to consider another person trustworthy [10, 23].

Several authors discuss and provide different attributes for trusting beliefs. They depend on the context. The most cited ones are from McKnight et al. [130]: benevolence, competence, honesty/integrity, and predictability. However, these beliefs focus on aspects of human social interaction and denote assessments that people make about each other.

Based on previous research, trust in inanimate software artefacts has been investigated under the premise that people perceive these artefacts as social actors, as if they possess human attributes. [126, 152].

In an extensive study in information technology artefacts, Lansing and Sunyaev [108] scrutinised the trust literature to identify trusting beliefs applied to inanimate software artefacts. From this perspective, an information technology artefact should be:

- Reliable: Perception that the artefact provides accurate content.
- Predictable: Confidence that the resource is always provisioned as requested.
- Provide the required functionality: perform as needed for the task environment.

In our study at hands, we reuse those three trusting beliefs.

A few studies in the software engineering field focus further on trust in artefacts. Reviewing empirical studies, Lansing and Sunyaev developed a conceptual model to describe trust in the cloud service software artefact. They comprehensively analyse the several trust constructs to reach a model contributing to theory-building on trust.

Koelmann [102] also provides a theoretical discussion around the concept of trust. However, he focuses on an information technology artefact's role in trust-related interactions. He further details the implications that can arise in situations involving trust.

Regarding empirical studies, we also found an original work focusing on trust in inanimate software artefacts. Thatcher et al. [186] examined how trust is affected by information technology and its support in users post-adoption. We consider this work as relevant to our study as it initiated the development of new concepts regarding trust constructs concerning inanimate software artefacts with evidence from an empirical investigation.

Paravastu et al. [152] explored how trust and trustworthiness apply to users' satisfaction. Their empirical investigation has shown differences between the two explored constructs, and user satisfaction is related to trustworthiness rather than trust in software artefacts. Trust in inanimate software artefacts is little examined in the software engineering field, especially regarding artefacts produced during the development process, which are utilised as resources from one phase to another.

As software development becomes geographically dispersed, people tend to rely more on artefacts to execute their tasks. To do so, they need to trust the information provided by the artefacts. In the interest of advancing research into trust in inanimate software artefacts, our study adds to our understanding of how trust can affect stakeholders' behaviour in software projects.

4.3 Research Method

Our study aims to examine the process of creating and utilising a BA. We are especially interested in the implications of trust beliefs and how they influence the stakeholders' behaviour in this process.

We developed our Research Questions (RQ) through discussions and brainstorming sessions with our company partner (introduced in Sect. 4.4.1). We addressed four RQs:

RQ1 How do practitioners decide how to represent knowledge and information in boundary artefacts and their expectations concerning their use?

RQ2 How do practitioners manage boundary artefacts?

RQ3 What are the implications of the differences in trust in boundary artefacts to software development?

RQ4 How does a difference in the level of trust in boundary artefacts influence the stakeholders' usage behaviour?

To answer these questions, we conducted a qualitative study analysing data on phenomena related to human behaviour [167]. We followed the principles of an exploratory case study to gather insights into creating and utilising the selected BA [160].

We collected the data via a series of sources, including interviews, archive documents, chat via Microsoft Teams^{TM} and a workshop. We examined the interviews by two-level coding (see Subsection 4.3.3) to identify relevant codes that enabled us to answer our RQs and explain the trust cycle (see Figure 4.6) [133].

4.3.1 Research Approach

Figure 6.1 depicts the approach that we developed to investigate the actions and interactions surrounding the BA. Our approach focused on three perspectives:

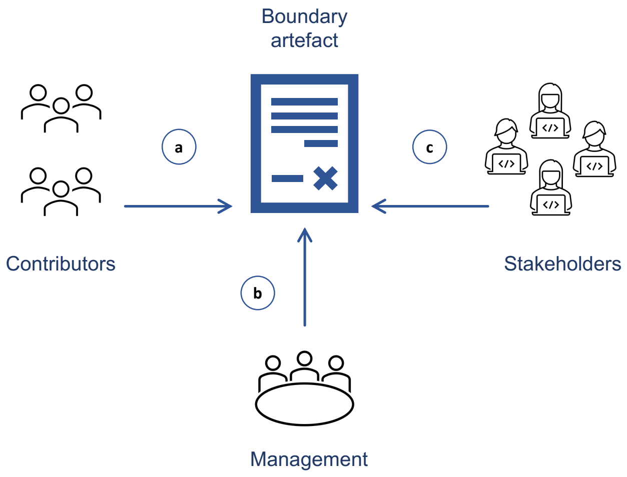


Figure 4.1: Research Approach

a. Contributors We investigated how the practitioners added content to the BA and their perception of who are the stakeholders of the object and how they utilise it.

b. Content management In this perspective, we identified content management aspects associated with the BA. We paid attention to existing guidelines for adding content, responsibilities, and ownership.

c. Stakeholders We explored how the stakeholders utilised the BA. We examined their perception of trusting beliefs in relation to the BA and how they influenced the stakeholders' behaviour.

This approach allowed us to identify misalignment between the content creation and management from the contributors' side and evaluate the trust of the stakeholders concerning the BA.

4.3.2 Data collection

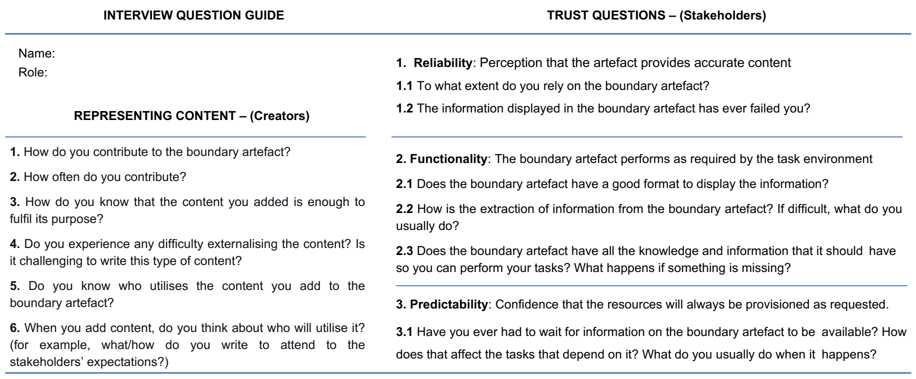
The data for this study was collected by means of interviews, archive documents, workshop, and informal conversations with practitioners. The first and fourth authors had full access to the company network and Information and Communication Technologies (ICTs) through dedicated employee accounts and devices set up and provided as part of the collaboration.

Table 4.1: Description of the practitioners included in this study (Interviews).

Participants	Role	Type
P1	Senior Engineer	Contributor
P2	Senior Engineer	Stakeholder
P2	Senior Engineer	Stakeholder
P3	Specialist Engineer	Stakeholder
P4	IT Product Owner	Management
P5	Experienced Engineer	Stakeholder
P6	Senior Engineer	Stakeholder
P7	Senior Engineer	Contributor
P8	Senior Engineer	Stakeholder

We utilised semi-structured interviews and one unstructured. The interview questions (see Figure 4.2) were selected according to the role of each practitioner who was interviewed, be it contributors or stakeholders (see Table 4.1) within the Quality Assurance unit, except for the content management approach. Thus, the inter-stakeholders differences are not of relevance to our analysis. In this approach, we used an unstructured interview due to the exploratory characteristic of the event.

We chose the trusting beliefs conceptualisation proposed by Lansing and Sunyaev [108] as our guide for elaborating the interview guide and analysing the results. It represents a combination of several theories of trust applied to inanimate software artefacts.



INTERVIEW QUESTION GUIDE

Name:
Role:

REPRESENTING CONTENT – (Creators)

1. How do you contribute to the boundary artefact?

2. How often do you contribute?

3. How do you know that the content you added is enough to fulfil its purpose?

4. Do you experience any difficulty externalising the content? Is it challenging to write this type of content?

5. Do you know who utilises the content you add to the boundary artefact?

6. When you add content, do you think about who will utilise it? (for example, what/how do you write to attend to the stakeholders' expectations?)

TRUST QUESTIONS – (Stakeholders)

1. Reliability: Perception that the artefact provides accurate content

1.1 To what extent do you rely on the boundary artefact?

1.2 The information displayed in the boundary artefact has ever failed you?

2. Functionality: The boundary artefact performs as required by the task environment

2.1 Does the boundary artefact have a good format to display the information?

2.2 How is the extraction of information from the boundary artefact? If difficult, what do you usually do?

2.3 Does the boundary artefact have all the knowledge and information that it should have so you can perform your tasks? What happens if something is missing?

3. Predictability: Confidence that the resources will always be provisioned as requested.

3.1 Have you ever had to wait for information on the boundary artefact to be available? How does that affect the tasks that depend on it? What do you usually do when it happens?

Figure 4.2: Interview guide.

The interviews were conducted through Microsoft Teams™ and lasted around 30 minutes. We also utilised Microsoft Teams™ for gathering data during the entire collaboration in informal chats with several practitioners. We initiated the data collection in September 2021 with the interviews and finished in April 2022 with the workshop and informal discussions with practitioners.

We gathered data from archival files, including Confluence™ pages, internal presentations, and descriptions of tools. We also collected data from a workshop planned for confirmation of the results and additional data.

The workshop lasted 1,5 hours and happened in the last days of the data collection. We had eight participants interviewed in the previous step and two practitioners who had a close interest in the subject (see table 4.2). The purpose was to validate the draft of the results and collect complementary data. We focused on the three aspects of our proposed approach (see Figure 6.1).

We stored three data types (audio, video and text) in the MAXQDA[1] software for qualitative and mixed methods research. The software supported us in extracting the codes in the two-step coding cycle directly from the data source.

[1]Available at https://www.maxqda.com

Table 4.2: Description of the practitioners included in this study - Workshop.

Participants	Role	Type
P1	Senior Engineer	Contributor
P2	Senior Engineer	Stakeholder
P3	Specialist Engineer	Stakeholder
P4	IT Product Owner	Management
P5	Experienced Engineer	Stakeholder
P6	Senior Engineer	Stakeholder
P7	Senior Engineer	Contributor
P8	Senior Engineer	Stakeholder
P9	Engineer	Engineer invited for discussion
P10	Expert Engineer	Experienced engineer invited for discussion

4.3.3 Data analysis

The data collection process and the analysis of the data were interrelated. The first author transcribed the interviews and established the initial codes. Simultaneously, the first author contacted people already interviewed and other practitioners in the company to clarify internal processes and misunderstandings.

We performed a two-cycle coding [133]. In the first cycle, we assigned process codes to the data chunks. This coding method indicates observable and conceptual action in the data and summarises data segments. The research questions partially steered the codes. For example, when the chunk of data referred to RQ1 and RQ3, we often coded it as "contributing to the boundary object" and "trust implication" (see Figure 4.3).

In the second cycle, we broke down the codes generated in the first cycle into smaller concepts that represented patterns in the data (see Figure 4.3). After concluding the second cycle, the first and the second authors gathered to analyse the generated codes. In this session, we discussed all codes and how and why they were added, aiming for consistency. We also examined the codes against the transcripts. We corrected misunderstandings, consolidated the initial findings through agreement, and identified gaps that required further clarification.

To increase reliability, we conducted a workshop with practitioners participating in the study. We asked for confirmation of our findings in relation to the three aspects of our approach, including content creation and management of the BA, the implication of trust beliefs, and practitioners' behaviour changes. After the workshop, we adjusted our interpretations regarding the content man-

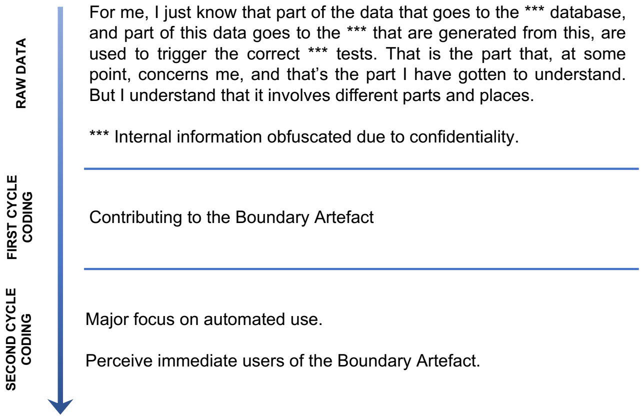

RAW DATA
For me, I just know that part of the data that goes to the *** database, and part of this data goes to the *** that are generated from this, are used to trigger the correct *** tests. That is the part that, at some point, concerns me, and that's the part I have gotten to understand. But I understand that it involves different parts and places.
*** Internal information obfuscated due to confidentiality.
FIRST CYCLE CODING
Contributing to the Boundary Artefact
SECOND CYCLE CODING
Major focus on automated use.
Perceive immediate users of the Boundary Artefact.

Figure 4.3: Example of the two-cycle coding process.

agement findings, which changed to *partial* content management. Although we validated most of the results, we still had open questions that required further investigation, including the stakeholders' behaviour, which was later clarified in discussions with practitioners through Microsoft Teams^{TM}.

4.3.4 Ethical concerns

We followed the Swedish Research Council [183] guidelines for conducting ethical research. Besides the guidelines, we also paid attention to how the interview guide was elaborated. We were careful in designing a guide that would not raise intense emotions or cause psychological harm [4].

The participants knew the study's goal and that they could choose to remove their interview from the research before the research was published. We asked for permission to record the interviews and explained that the data would be stored on the company's computer only. Moreover, every mention of specific pieces of data, such as quotes, would be anonymised.

When sampling the practitioners to participate, we decided based on the following criteria: their role (creating content, managing or utilising the BA) and availability during the period we performed the interviews.

We had an intermediate step in our research where we conducted a workshop to present a draft of the findings, aiming for validation and additional data collection. During the workshop, we were cautious not to reveal participants' names or the teams they belonged to, avoiding any discomfort.

Regarding the data analysis, we avoided potential stigmatisation or harm to specific populations by not collecting or analysing data related to the religion, race, or ethnicity of any participant in this study.

4.4 Results

In this section, we report on the results of our study, which investigates the creation and utilisation of one specific BA and the implications of trust and their influence on stakeholders' behaviour. We divide the subsections in tune with our RQs (see section 4.3).

4.4.1 Case company

Axis Communications provides network solutions in video surveillance, access control, intercom, and audio systems. They collaborate with thousands of technology and system integration partners around the world to deliver solutions and support customers with focus on openness and quality. Axis has around 4,000 employees in over 50 countries and has its headquarters in Lund, Sweden, which is where the case study was centered.

The AXIS OS operating system for edge devices is used in more than 200 of Axis' products. Over 1,000 research and development engineers work directly or indirectly with developing AXIS OS as a software product line. There are also other Axis employees and Axis partners that are dependent on information about individual products in the AXIS OS product portfolio, such as technical writers, support personnel and development partners.

4.4.2 Description of the boundary artefact

The BA analysed in this study represents the core list of features of all of the products in a software product line of the company (covering over 200 products), and it is physically represented as an XML file. It stores all the developed features for the firmware used in the company's products, be it new, modified or deprecated features. Teams with specific functions (approximately 48 teams), e.g., audio and streaming, add the features and associated content to the BA.

The artefact is handled as code and managed in the same source code management tool used for the product code for the software product line. Automated regression tests for the software product line use the BA so that discrepancies are detected in the build pipeline. Those that need changes to the BA will create pull requests, represented by the dotted lines in Figure 4.4.

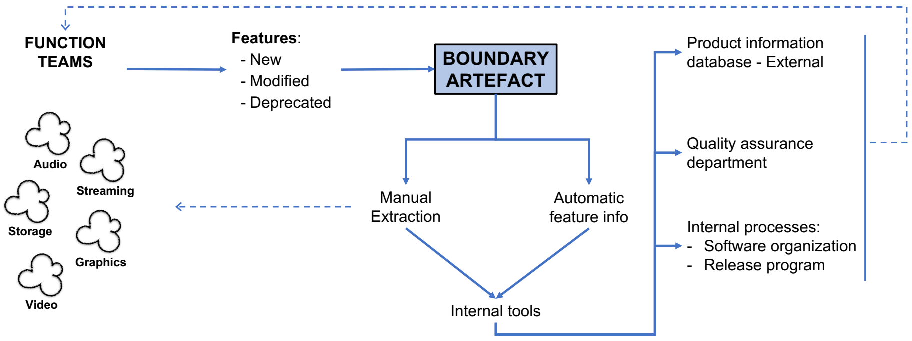


Figure 4.4: Depiction of the utilisation of the boundary artefact

Figure 4.4 also displays a summary of the several internal stakeholders in different parts of the organisation. Customer and external partners (for example, application development partners) are also stakeholders. Our investigation focused on the Quality Assurance department.

The content of this BA is utilised in automated and manual formats, but mostly automated. The information is automatically retrieved through a framework that collects product information using an API (Application Programming Interface).

Through the automatic retrieving, information from the BA crosses several boundaries within the organisation, including marketing, global services, quality assurance, platform management, solution management, customer information, etc. At each boundary, the information provided by the BA serve different purpose.

Besides the mentioned department boundaries within the organisation, the BA represents a shared structure for teams with similar activities but different goals and focuses. Those characteristics resemble the *Coincident boundaries* artefact type described by Star and Griesemer [175] (see Section 6.2). It has the advantage of providing input to perform tasks with different goals. Teams work autonomously but collaborate in this shared artefact.

4.4.3 Representing and Managing the Content of the BA

In this subsection, we detail the process of adding content to the BA and also the approach adopted to manage it, addressing both *RQ1* and *RQ2*.

So-called function teams add new features and make adjustments to existing ones. This process is meant to be flexible. Creators in such teams begin with rudimentary versions of the content and improve them until they have a satisfactory version, such as descriptions and parameters.

Regarding the new features, the content addition is guided by a multidisciplinary team - having the goal of building a generic vocabulary that the whole company can use. They book sessions so the creators can explain the new feature. The team then choose names for the features and make sure that the names are technically correct and easy to translate. The multidisciplinary team partially manages the artefact with the activities detailed in Table 4.3.

Table 4.3: Activities for managing the BA.

Activity	**Description**
Obtaining an overview of the utilization	Awareness of how the content is used but predominantly in the automated side and generation of information to customers.
Assigning responsibility for parts of the content	Finding owners for each feature.
Achieving flexibility in adding content	Possibility for incremental edits to the existing content by the teams.
Capturing knowledge about each feature	The multidisciplinary team establish meetings with creators for details about new features.
Establishing a common terminology	Defining terms and names to be known by the whole company and customers.
Planning for future increments	Predicting that a feature has the potential to grow and list the future options to reduce the updates.
Controlling over modifications	Every modification generates an approval request which at least two people review.
Creating generic content	Making features for different setups and not connected to a specific device.

The management team also clear up the features that do not belong to the artefact and deprecate features that are no longer in use but keep there as historical data.

In our investigation, we found that those function teams perceive the immediate users of the artefact. Their focus is on automated use, while the manual extraction of the content is not widely known, exemplified by P1: "*For me, I just know that part of this data goes to the database, and part of this data goes* *to the feature lists that are generated from this, are used to trigger the correct tests."*

As this happens, the stakeholders of the manual extraction do not have their needs observed. We detail these findings in the following subsection and the discussion section.

4.4.4 Implication of Trusting Beliefs

In this subsection, we detail our investigation of the differences in trust and the implications generated when the BA is not reliable or predictable - answering the *RQ3*.

As the stakeholders rely on BA to execute their tasks, they expect the information stored in the artefact to fulfil their needs. Several implications can affect the development process when practitioners understand that information does not meet favourable factors of trusting beliefs.

The information provided by the BA meets the **functionality favourable factor**. One explanation is that most of the content is in a code format, read by developers and other tools that automatically extract the information. In this aspect, P2 commented: "*Well, the XML is never human-readable. So, no, it doesn't, but programmatically it works, of course. There is no problem in reliability for the code in the BA, but it is not human readable.*" To a stakeholder not involved in this level (development), a web version of the artefact displays the description for each feature. For those stakeholders, the web version meets the functionality factor.

As to **reliability**, we found that the content stored in the BA failed the stakeholders. When the stored information is incorrect, the automatic extraction can run tests that will fail, execute wrong tests, or not even run them. P8 exemplifies saying: "*It sometimes affects when we execute new tests. The BA must be updated before the tests run on devices with the correct features.*" P7 adds, "*There have been updates to the BA that broke tools that I used because it didn't follow the XML standard or someone added something that simply didn't work. So, there failed from time to time.*"

When those issues occur, it is evident that the development process can suffer a setback due to waiting time to fix them. The incorrectness of the information can also generate mistakes in appointing tasks to the wrong practitioners. As we mentioned in the previous section, when the content creators of the BA are not familiar with the stakeholders' needs, the content generated won't support the execution of their tasks. In this regard, P6 stated: "*I think it is important that the features have the right ownership to address tasks concerning that features* *to the right tester. The information can be fetched through other sources, but it takes longer, so the BA is a quick way to get it.*"

Terminology occasionally causes misunderstanding because, depending on the department of the practitioners, they spend some time understanding what the feature is about. Regarding this issue, P2 commented: "*I need to land in the BA first to understand which feature I am looking for because the namings in the BA are not the same as the namings that I'm used to. Because it is not the same name I use in software terminology, sometimes it is confusing to figure out how a feature is activated on the camera.*"

Regarding **predictability**, we examined if the practitioners had to, at any time, wait for information to be available. When the information of the BA does not meet this criterion, the automated tests get delayed, which connects to the implications found in the reliability factor. In this regard, P4 explains: "*If it is missing, you sure have to wait. I am not the one who put it in. So, I have to tell someone who has to go through this whole process, so there is a huge delay. I expect it to take a long time if it is not in there.*"

At last, the lack of predictability generates the feeling of uncertainty about the availability of the information by stakeholders. As P3 observed, "*It sometimes affects when we can't take in new tests. The BA needs to be updated before, so the tests are run on devices with the correct features.*"

The combination of the reported implications influences the stakeholders' behaviour as they need to execute their tasks. We describe this behaviour in the following subsection.

4.4.5 Change in Behaviour

In this subsection, we explore how a difference in the level of trust in the BA influences the stakeholders' behaviour *(RQ4)*.

There are features with multiple implementations in the BA due to differences among the products. In some situations, the stakeholders find it difficult to know if they can use one of those implementations. They can't use it in other cases because the BA does not offer the necessary implementations.

To execute their tasks, the stakeholders create workarounds such as complementary artefacts. They are made within the function teams and address their specific need. This behaviour is derived from our reliability investigation. When the information fails the stakeholder, they move towards another solution that won't be ideal, which is this case. Note the remark made by P4: *"It is often that we can't use the BA in those cases, so we can't make a test out of it. We have our own test suite with our own made-up BA for many products."*

The content is expected to be merged with the official BA. However, there is no control over when those cases happen, how many teams create them, or if the content has been actually merged. P4 details this process: *"This artefact is official in the sense that we use it for our tests. No other usage than for test purposes. Just my team, as far as I know. It grows with the need to separate tests for products. We add that if we have a test that needs to be run depending on a feature. If the official BA had these implementations, we would not need it."* During the workshop, we also confirmed that this happens to other teams, which can be disastrous, according to one of the experienced engineers. Besides, during our investigation, one feature was missing in the BA.

To better visualise our research findings, we summarized them in Figure 4.5. It shows the process of creation, management, and the implications of the differences in trust in a software project. Also, how they influence stakeholders' behaviour.

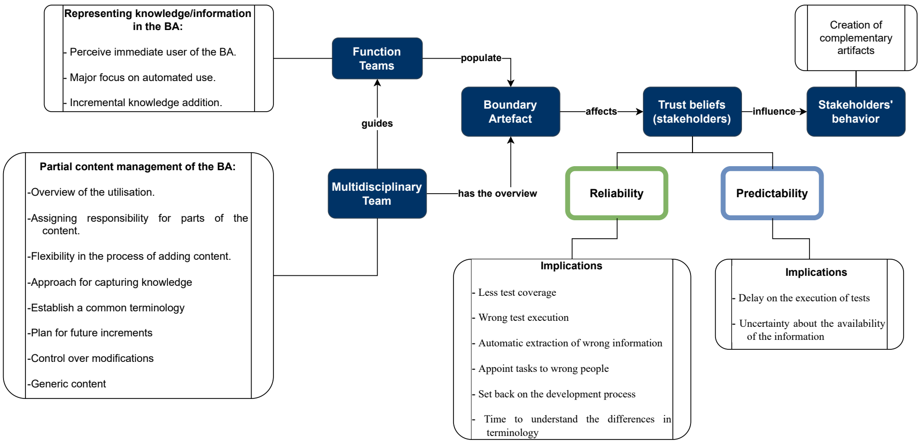

Representing knowledge/information in the BA:
- Perceive immediate user of the BA.
- Major focus on automated use.
- Incremental knowledge addition.
Function Teams
populate
guides
Boundary Artefact
affects
Trust beliefs (stakeholders)
influence
Stakeholders' behavior
Creation of complementary artifacts
Partial content management of the BA:
-Overview of the utilisation.
-Assigning responsibility for parts of the content.
-Flexibility in the process of adding content.
-Approach for capturing knowledge
-Establish a common terminology
-Plan for future increments
-Control over modifications
-Generic content
Multidisciplinary Team
has the overview
Reliability
Predictability
Implications
- Less test coverage
- Wrong test execution
- Automatic extraction of wrong information
- Appoint tasks to wrong people
- Set back on the development process
- Time to understand the differences in terminology
Implications
- Delay on the execution of tests
- Uncertainty about the availability of the information

Figure 4.5: Summary of the Findings.

Together these results provide important insights into how differences in trust over BA can affect software projects, more importantly, how they influence stakeholders' behaviour when using the BA. In our case, this behaviour can be detrimental because the complementary artefacts can remain unknown and possibly needed by other stakeholders. We believe that stakeholders might have other behaviours in a different context and use the BA in other ways or not even use them. We discuss the findings and potential solutions in the following section.

4.5 Discussion

This study aimed to understand the implications of differences in the perception of trust in BAs on software projects and their influence on stakeholders' behaviour. Our findings suggest that most negative implications and changes in stakeholders' behaviour occur because of inefficient management of the BA.

By abstracting the case results, we can establish a conceptual idea of how the trust cycle of boundary artefacts functions. We state that trusting beliefs about the BA influence how the stakeholders use the BA (see Figure 4.6). Negative implications might emerge when practitioners do not perceive the BA as favourable in at least one of the investigated trusting beliefs, reliability, functionality, and predictability.

Thus, the trust cycle is generalisable to different software development companies with different sizes. The negative implications resulting from the inconsistencies, though, can be the same or others because they are conditional to the context. For example, we did not find implications regarding the functional factor. Other contexts or different BAs can reveal them—also, different behaviours.

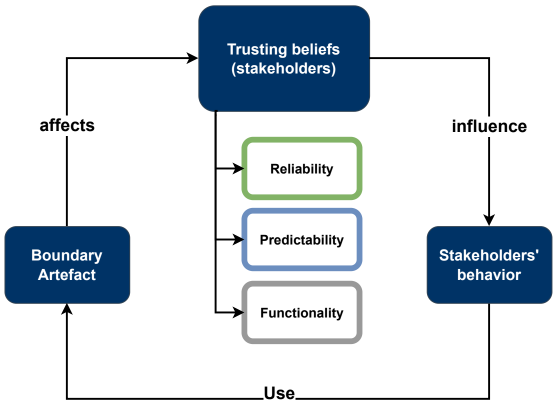


Figure 4.6: Trust cycle

As the BA carries many inconsistencies, the stakeholders can show less confidence, reducing their trust in the BA. They will use the BA differently from how it was supposed to be used or not use it all. For that reason, building up the value of the BA by increasing its trust levels is critical when driving positive behaviour from stakeholders.

In our case, the partial control over the BA has caused several negative implications and influenced the stakeholders' behaviour in creating complementary artefacts. Given that there is a constellation of different teams contributing to and utilising the BA, we believe that adopting a cooperative approach where the management of the BA is shared between practitioners is a good practice for managing the BA and keeping it trustworthy. Dividing the management work can reduce the workload compared to leaving the job to one or a few individuals.

Also, the studied BA already has partial management of the content. Our suggestions lie in the distribution of the work for the whole artefact. One example of this approach would be to divide the handling of the BA between the content and the technical owner when having machine-readable artefacts.

To collaborate with this approach, we provide potential solutions on two main levels. The first refers to the scope of the multidisciplinary team that currently owns the BA. The second level concerns the creation of the BA, which aspects should be observed.

Management Level

Having a **formal strategy for managing the entire content** prevents inconsistencies in the BA. This process can be done by assembling interconnected actors and defining roles and responsibilities for each part of the BA's content to provide correct and timely information [119, 202].

The incorrect information has caused complications to test execution, be it manual or automated. Every time spent to fix those punctual issues with the BA can generate avoidable loops and increase costs for corrective procedures [33, 123].

Such complications can be minimised by **periodic evaluation and feedback processes**. Stakeholders can evaluate the BA regarding the trust aspects through a questionnaire, for example. In addition, the creation of ways to receive and implement feedback can avoid reliability issues and prevent the need for accountability due to outdated or wrong content.

Content Creation Level

The contributors of the BA have limited knowledge about the stakeholders and the different usage of the content provided. Yet, one crucial aspect of the BA is that the content represents its stakeholders' needs and is a fundamental part of the BA's design. We believe **mapping the stakeholders, and their needs** is a key point for designing BAs. It requires negotiations between contributors and stakeholders to reach a consensus about the multiple domains of stored knowledge [2, 87, 98].

Occasionally, terminology causes confusion and misinterpretation. When building complex products, the amount of knowledge and its types are often large, creating different terminology levels. The literature reports three main **approaches for dealing with terminology** in BAs, which vary depending on the terms' novelty level [33].

- Syntactic boundary: usually, a common lexicon is sufficient to address the differences—other techniques include taxonomies and store and retrieval technologies.
- Semantic boundary: there is a need for common meanings to create shared meanings to address the differences and dependencies that originated from the terms' novelty. Other techniques: cross-functional interactions/teams and boundary spanners/translators.
- Pragmatic boundary: demand for artefacts that can be jointly transformed. Prototyping is a technique to be used.

Accommodating Experimental and Evolutionary Content is essential for the case we focus on. It seems counter-intuitive to have such flexibility when BA are highly structured from its original characterisation. However, software development is a dynamic activity that usually welcomes change while developing and producing experimental content.

In the case we focus on, there is a frequent need to test new features that are not yet implemented; thus, it is not in the formal BA. Therefore, it is important to be flexible to accommodate this type of content that can be changed or removed. Otherwise, the content will remain within the teams and probably not be known by others. In addition, as technology evolves, the need to create content in different formats also changes. For that purpose, **foresee how new content adapts** becomes valuable.

Our study revealed how the differences in trust in BAs affect software projects in different areas of the organisation and interfere with the task execution of various stakeholders. The decrease in trust results from inconsistencies in the content associated with the lack of management of the BA. A structured strategy for representing and managing a BA's content seems appropriate to increase trust levels and efficiency.

It should be noted that we have not investigated the impact of the negative implications on the software project. Although we revealed them, we do not know how severe they are. A future study could examine how to measure the implications' severity level. The results could support practitioners in prioritising activities for making adjustments and fixes. This is particularly important when dealing with large and well-established BAs that would require an excessive workload for restructuring them.

Another aspect we believe is important for better-designing BAs is investigating the notions of value for different stakeholders. A qualitative study exploring the stakeholders' perceptions of value and how to utilise them could shed light on creating BAs. The nuances of value can differ and provide different aspects that can offer additional value to the stakeholders rather than only the correct content, for example. We believe that if the artefact is well-planned from the beginning, there is a chance that fewer issues will occur regarding how the artefact supplies the stakeholders' needs.

Lastly, the third research direction. We suggest replications of this study to contribute to building a body of knowledge on implications to software projects when practitioners do not perceive the BA as favourable concerning trusting beliefs. Other cases could help us classify negative implications by company size, number of stakeholders, BAs not machine-readable, etc.

4.6 Threats to validity

In this section, we discuss a number of threats to the validity of our study. We followed the guidelines recommended by Runesson and Höst [160], who classify such threats as a construct, internal, external, and reliability threats.

Construct validity refers to how the theoretical constructs investigated are represented by what the researcher has in mind and what was actually investigated during the interviews. In this aspect, we identified one threat: the interpretation of the criteria to evaluate trusting beliefs such as reliability, functionality and predictability. To reduce this threat, during the interviews, we defined each criterion. For each of them, we had questions that helped the practitioner understand them in the context of information in the artefacts (see Figure 4.2).

The internal validity concerns whether a third factor might be affecting the factor in the investigation besides the ones already identified. As we chose an exploratory study, this threat is minimised because we do not use pre-defined concepts to be confirmed. To diminish this threat, we utilised a semi-structured interview guide that allowed the practitioners to speak spontaneously. In addition, we validated our findings in a workshop that worked as both a sanity check and additional data collection.

External validity relates to the extent to which it is possible to generalise the study's results to other cases. Also, to what extent do the findings stimulate the interest of other practitioners. As a case study, our results provide analytical generalisation through plentiful contextual information and discussion of the findings [66, 160]. The findings of this study extend to cases with common characteristics, such as companies that develop hardware and software and utilise a BA to store and distribute information about all features throughout the development process.

Concerning the reliability threat, we discuss potential threats related to how the analysis depends on a specific researcher. To minimise this threat, we provided an extensive description of our research approach and a detailed explanation of the coding process with examples of the raw data. Also, we offered the semi-structured interview guide as a transparency action.

4.7 Concluding remarks

Boundary artefacts support task execution for many stakeholders within a software development project. However, if practitioners do not trust the artefact, they will probably not rely on them or have different behaviour toward them.

We conducted an empirical investigation through a case study to reveal how people create, utilise and manage BAs. Moreover, we observed how differences in trust affected software projects and how they change the stakeholders' behaviour.

Our investigation has shown that practitioners who add new and adjust existing content do not entirely understand the stakeholders' needs. In addition, the management of the content is partial. These findings indicate that parts of the content and stakeholders' needs are not supervised, impacting trust levels.

When practitioners do not perceive the BA as favourable in at least one of the investigated trusting beliefs, specifically reliability and predictability, the stakeholders can't execute their tasks appropriately, and several implications affect the software project development. Additionally, the stakeholders create workarounds to supply their needs. Among those, they create complementary artefacts within their teams.

To function as planned, organisations need to increase the level of trust of their BAs, and the proper management of the content is crucial for increasing this trust level. This management should include creating content based on stakeholders' needs and monitoring changes. Future research could explore and develop other ways for supporting practitioners in creating and managing such artefacts in software development environments.

Acknowledgements

We want to thank all practitioners at Axis who participated and contributed to this study. We would also like to acknowledge that this work was supported by the KKS foundation through the S.E.R.T. Research Profile project at Blekinge Institute of Technology.

Chapter 5

Thinking Strategically about Knowledge Management in Agile Software Development

This chapter is based on the following paper:

> Raquel Ouriques, Krzysztof Wnuk, Richard Berntsson-Svensson and Tony Gorschek. "Thinking Strategically About Knowledge Management in Agile Software Development." In Kuhrmann M. et al. (eds) *Product-Focused Software Process Improvement.* PROFES 2018. Lecture Notes in Computer Science, LNCS 11271, 389–395, (2018)

5.1 Introduction

The knowledge of individuals is a well-known competitive resource, and has been philosophically discussed and validated by several researchers within the strategic management field [11, 50, 140, 154]. Knowledge Management (KM) strategies, which are based on knowledge needs and organizational characteristics [23] [161], help with effective knowledge resources management. Successfully employing the knowledge resource contributes to product and company growth [185].

Software development is significantly dependent on exploiting the knowledge resources [18, 161]. However, the Agile Software Development (ASD) empirical literature seems to give more attention to descriptive studies that report the use of tools and practices for knowledge sharing, rather than their effectiveness and impact on the software development [18, 148].

We identified in our previous study that the KM practices reported in empirical studies in ASD context have low or no connection with the strategic level of the companies, which has negative implications for traceability between the development practices and the company strategy, and measurement of successful implementation of these strategies [148].

Treating knowledge as a resource implies that it needs to be managed with a logical and structured approach [161] [11]. We conjecture that the lack of integration of KM practices with the strategic level, together with the adoption of informal KM practices adds low value to the companies that adopt ASD, and hinders unlocking the full potential that ASD brings.

Based on the recent results of our literature review on empirical studies [148] and on KM theories from the strategic management domain, we discuss future research implications of considering knowledge as a resource in ASD.

5.2 Knowledge Management in Strategic Management Domain

The rationale behind using theories is the glue that connect the explanatory factors [195]. Therefore, we base our discussion on concepts originated in underlying theories of the strategic management domain.

The theoretical literature in the referred domain states that the knowledge embedded in organizational routines contributes to company's effort to achieve its goals, by creating and delivering value to its customers [103, 156]. The value created can be product value as well as organizational competence in solving problems and addressing customer needs.

The value creation process has social and cognitive relationships for enhancing individual's abilities for production. Therefore, organizations arrangements tend to change regarding hierarchical aspects, varying from the factory model to flat organizations. In these different structures, activities are more or less knowledge-intensive, which influence how individuals create value [125].

Several companies perceive the strategic relevance of knowledge as the main competitive resource, especially in knowledge-intensive environments. However, building and maintaining the competitive advantage based on the knowledge resource, requires a strategic plan that integrates the long-term vision with the knowledge resources [75].

Companies are often affected by changes in the external environment, e.g., suppliers, market, laws and customers' requirements. In this changing environment, companies need to adapt themselves in order to be more responsive and monitor current and future knowledge needs.

In previous work, we envisioned a conceptual framework (refer to Fig.5.1), that displays a high-level perspective of the integration between the corporate strategy and the project level of ASD. The KM strategy reflects the corporate strategy, and promotes, through practices, one or more knowledge processes (KC - knowledge creation, KA - knowledge application, KT - knowledge transfer, KS - knowledge storage) through the company's levels.

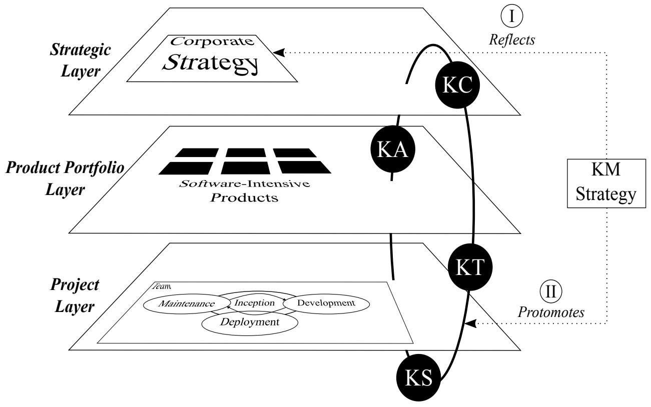


Figure 5.1: KMS-ASD [148]

The corporate strategy guides how a company acquires and distributes its resources, e.g., if the company aims for product innovation, which investments in technology and knowledge should be made. In each level, KM strategies might influence different knowledge processes, depending on knowledge needs and their sources [148].

In a study in the manufacturing and services sectors, Ferraresi et al. [65] provided empirical evidence that the business performance is positively impacted by KM strategies only when these strategies are connected to a strategic orientation. Product quality is also affected by KM. Lee at al. [38] investigated in several industry sectors how KM is linked to product quality. They found that product quality is significantly affected by how process management moderates customer knowledge acquisition and the participation of employees in KM activities.

KM effectiveness seems to have a relation to the management encouragement. The rationale behind a KM implementation involves an internal analysis of the company's resources and its needs. Then, the decision regarding which tools and methods to use to achieve the selected goals need to be made [23].

The literature gathers three common approaches for conceiving KM strategies: The rational approach - considers an analysis of the companies resources and need, and external environment of the company, such as market and competitors. The emergent approach - developed based on the employees' daily activities, for example, their methods of problem-solving and their knowledge needs. The integrated approach, which is the combination of both rational and emergent approaches - it is a dynamic interaction where the strategic level provides guidance with the company's general vision, supported by inputs from the lower level [23].

In any of the three common approaches, we notice that the connection with the corporate strategy of the company is recurrent. Another important aspect that should be considered is the domain where the KM strategies will be executed, by reason of companies size and hierarchy, and knowledge intensiveness of the activities.

5.3 Knowledge Management Strategies in Agile Software Development

Most of the software development activities are knowledge-intensive [162] [54], which evoke the need for the companies to make efforts towards leveraging individuals' competence throughout the organizational layers. However, when it comes to KM practices in ASD, there is a lack of connection between these practices and the corporate strategy [148].

In a previous literature review on empirical studies [148], we mapped the KM practices in software companies adopting ASD (Refer to Figure 5.2) and which type of strategy these practices follow. Most of KM practices in ASD are focused on personalization strategies, which focus on human interaction to communicate knowledge, and the majority of them are established in the project layer.

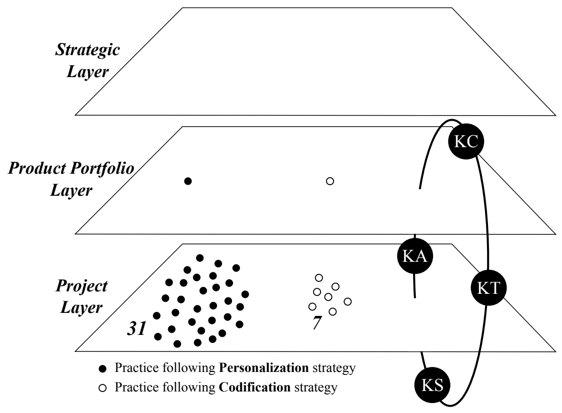

Strategic Layer
Product Portfolio Layer
Project Layer
31
7
KC
KA
KT
KS
● Practice following **Personalization** strategy
○ Practice following **Codification** strategy

Figure 5.2: Distribution of KM practices in organizational layers through KMS-ASD [148]

Moreover, there is a tendency of isolation regarding agile teams when it comes to KM, see Figure 5.2. Most of the KM practices might work well inside the teams; however, these practices are informal and occasional. The lack of connection between KM practices and the corporate strategy can result in deviations from the core vision, wasted resources and irrelevant knowledge acquisition that adds no value to the company [11, 24, 161].

The corporate strategy guides the resources allocation for achieving organizational aims [23] [148]. In the resource-based view of the firm, Barney [11] emphasizes that the means of how a company allocates and uses its resources confers the competitive advantage that differentiates the company from its competitors.

Knowledge is an asset that requires comprehensive and logical management to obtain benefits, since, together with skills, it is the main resource of software development organizations [161]. Steen [176] argue that in software development, the product quality phenomenon is heavily dependent on knowledge and skills, however, few research explores this relation.

Santos et al. [164] observed, in an empirical study, that in an ASD context, knowledge sharing effectiveness has relation to purposeful practices, organizational conditions and the stimuli that individuals have to share.

Similarly, goal-modeling may be used in the beginning of software projects to align the system to the organizational goals, which also might indicate that goal-oriented practices are a positive approach to further effectiveness measures for KM; why the practices are needed for; how do them connects to corporate strategy and customer needs.

Both empirical findings and theoretical dialogue connect our discussion, within which the strategic management aspects of having knowledge as a resource, have implications for KM strategy planning.

5.3.1 Potential Research Opportunities

Since 2010, publication related to KM in ASD gained diversity regarding KM focus, e.g., practices, challenges, and theories. Despite that, the state of the research remains far from the KM mainstream in strategic management studies.

We highlight the following research opportunities for exploring KM in ASD:

- ***Strategic KM***. Concerns regarding KM effectiveness were raised in previous studies in software engineering [18] [148]. Knowing that KM practices produce the desired results might be crucial to a company on deciding to invest in KM. The two essential aspects to consider in planning KM strategies for a company are the connection between the corporate strategy with the organizational arrangement; and the long-term goals of the KM strategy. [23]. Illustrating these elements in ASD contexts, we could explore: how KM strategies comply with coordination? what adaptations are necessary? Could goal-oriented KM practices facilitate KM effectiveness measurement in ASD?

- ***Product quality***. A KM resource remains valuable to the extent that it can deliver value to the customer, and also contributes to achieving enhanced performance [185]. Empirical research has shown that the degree of participation of an employee in activities related to knowledge dissemination impacts the quality of new products significantly [203]. Steen [176] found that software product quality cannot be entirely formalized, but rely on, to a great extent, the practical knowledge, and experience of individuals. Since knowledge is the main resource for software development, future research should keep attention on how to manage the knowledge resource, in ASD context, in a way that it provides superior customer value. What KM activities result in better product quality? In what context? How these activities affect quality in the software development process, such as requirements, implementation, testing, validation and verification?

5.4 Conclusion

In this vision paper, we discuss the potential research opportunities of KM perspective in ASD. Our inference is that we have reached a degree where the research demands investigations that go beyond mapping the companies actions against the KM theories, to real planned interventions with companies. Knowledge is socially created and translated into processes and products, representing unique characteristics that every company has. Future research should aim to gather more empirical evidence regarding effectiveness, and substantial impacts of KM strategies in coordination and other aspects, such as software quality.

Acknowledgements

The work is partially supported by a research grant for the ORION project (reference number 20140218) from The Knowledge Foundation in Sweden.

Chapter 6

Preliminary Guideline for Creating Boundary Artefacts in Software Engineering

This chapter is based on the following paper:

> Raquel Ouriques, Fabian Fagerholm, Daniel Mendez, Tony Gorschek, and Baldvin G. Bern. "Preliminary Guideline for Creating Boundary Artefacts in Software Engineering". Submitted to *Information and Software Technology Journal (Under review)*, IEEE, (2023).

6.1 Introduction

Boundary artefacts (BA) are objects crossing the boundaries of social worlds, carrying several meanings that meet the information needs of different stakeholders [175]. Software development utilises BAs to a large extent and for different purposes, including requirements specifications, architectural descriptions, and project planning documents [12, 206].

Software development benefits from BAs by supplying different teams with condensed information in several formats. Also, these artefacts provide the means for keeping and carrying relevant information helping coordination among several stakeholders. For example, software requirements specifications are used by, and thus support, different roles such as architects, project managers, and stakeholders at customer side, by conveying information that has different purposes for each role across each of the boundaries.

While BAs strengthen collaboration by providing timely and correct content throughout development projects, inconsistencies in BAs can affect how practitioners perceive and utilise them [21, 145].

The organisation of BAs affects the trust that practitioners deposit in them and also influences their behaviour when utilising those artefacts [145]. When practitioners do not feel confident about BAs, their level of trust decreases. Besides the effects on utilisation, several negative implications for software projects can emerge, such as a setback in the development process, uncertainty about the availability of the information, time wasted looking for information and wrong execution of tasks [145].

Decreased trust levels can also influence practitioners not to use the BAs and re-create their solutions, causing frustration. Such inefficiency in managing companies' knowledge through artefacts can result in increased costs and time waste [147].

As trust is an essential factor guiding the utilisation of BAs in software projects, there is a need to understand which principles should be observed when creating such artefacts and to what extent these principles contribute to creating trustworthy artefacts [145].

Although there seems to be a common understanding of the vast production and use of artefacts in software engineering, to the best of our knowledge, the community still lacks necessary guidelines required to support an effective creation and management of BAs. We contribute to filling this gap by investigating principles that support the creation of trustworthy BAs.

In this manuscript, we report on the following contributions:

- We develop preliminary guideline focusing on the creation of BAs.
- We conduct an expert validation with a company partner to validate our proposed guideline and implement their feedback.
- We collect the participants' perceptions on how the proposed guideline contribute to creating trustworthy BAs and their usefulness.
- We provide supplementary material for transparency and potential re-application of the guideline in other contexts.

This manuscript is organised as follows: Section 6.2 presents a brief background to our study and related work. Section 6.3 describes our research methodology. Section 6.4 presents the developed guideline and the results of the expert validation. Section 6.5 provides a discussion of the results and implications that our current findings have for future research. In Section 6.6, we discuss the lessons learned after our validation. Section 6.7 describes the threats to the validity of our study. Lastly, in Section 6.8, we present our concluding remarks.

6.2 Background and Related Work

In this section, we introduce the basic notion of BAs and how trust relates to inanimate software artefacts as the background of our study. We conclude by discussing related work.

6.2.1 Boundary Artefacts and Trust

The term boundary artefact (often referred also as boundary object), was originally coined in a study by Star and Griesemer in the Museum of Vertebrate Zoology [175]. We adopt the term boundary artefact here due to its closeness to the software engineering. The authors refer to the term as an object having enough flexibility to adapt to local needs, providing shared understanding among different users while keeping its identity as they cross many boundaries. In their work, the authors offer the example of a university administrator world who is often involved in contracts and grants, performing a particular set of tasks and dealing with different audiences. At some point, the administrator's world intersects with the world of a naturalist, in which part of the job is to collect specimens for a natural history museum. When this happens, the boundary object will minimize the difficulties with communication and cooperation by providing content that can satisfy their needs even though having different meaning in different worlds.

Boundaries are not always synonyms with a limit in organisations. Instead, they are mostly as permeable. That is, they are not stoppers of information in an organisation, but there is a transition where people on one side of the boundary see information from their perspective, while people on the other side see it from another perspective – and they might be looking at the same information or working towards the same goals. For example, a use case developed to analyse requirements will be utilised in many other software development activities such as project organisation and testing. Every time the artefact crosses a boundary, its purpose can change depending on the people's needs.

Boundary artefacts have a few characteristics that help distinguish them from other types of artefacts. They can provide shared meaning by satisfying people's needs in different social worlds. To be able to do that, they are plastic enough to meet local needs and keep a common identity when crossing boundaries. When in common use, they are weakly structured but become strong in individual use due to their ability to meet local needs. Their representation can be abstract or concrete [175].

Software development benefits from having BAs as a single artefact can supply stakeholders in different boundaries, especially in geographically distributed environments. However, when lacking proper management, these artefacts can be anywhere from being overloaded with meanings that render it difficult to find relevant information to the absence of enough details. In such a situation, BAs can lose their value to stakeholders and decrease their efficacy [2, 201, 206].

Intergroup politics around BAs can affect how stakeholders utilise them. They are not always politically neutral, and their subjective dimension often makes them dependent on trustworthiness [88, 99]. Trust plays a relevant role in driving the positive behaviour of stakeholders towards BAs.In inanimate artefacts, trust has been explored under the premise that people perceive inanimate software artefacts as possessing human attributes [126, 152].

There is no consensus about a definition of the term, but for a general understanding, trust refers to how individuals deliberately rely on others [190]. Due to its abstract characteristic, it is complex to assess. To address this complexity, trusting beliefs have been developed to help describe the favourable factors that make individuals trustworthy, such as benevolence, integrity, predictability, and competence [128, 130, 190].

An extensive study on information technology investigated which trust beliefs best describe trust in inanimate software artefacts [108]. The authors conducted a literature review and identified the following trusting beliefs: Reliability - Perception that the artefact provides accurate content. Predictability - Confidence that the artefact content is always provisioned as requested. Provide the required functionality - perform as needed for the task environment.

Trust in inanimate software artefacts is little examined in software engineering [145], especially principles that contribute to creating trustworthy BAs. Principles are dispersed and not examined from the trust perspective.

6.2.2 Related work

Few studies provide solutions or practices targeting BA in software engineering. Radhika et al. [91] investigate BAs' roles in addressing requirements engineering challenges when having multiple stakeholders in the product family development context. Through a multiple case study, they provide insights on utilising BAs to improve the quality of requirements engineering processes and suggestions for managing such artefacts.

Focusing on the automotive industry, Wohlrab at al. [201] created practical guideline for managing systems engineering artefacts. They analysed each type of artefact's challenges and provided solutions that were further packed into contextualised guideline.

Loggem and Veer argue [189] favouring a documentation-centred approach to support internal and external communication (focusing on users). According to the authors, the BAs potentially solve two main issues, heterogeneous mental models within teams and the peripheral role of the user concerning the teams.

We have previously investigate how differences in trust level can affect stakeholders' behaviour towards BAs [145]. When the content of the BAs does not meet stakeholders' needs, several implications affect the software development project. To minimise the implications, the authors offer potential solutions.

Although the referred studies offer suggestions for practices regarding the creation and content management of BAs, the practices are particular to specific cases. We believe this is the first work that gathers generic principles into preliminary guideline that support the creation of BAs in software engineering. In the interest of advancing research into trust in inanimate software artefacts, this study adds to our understanding of how the guideline' principles can help create trustworthy BAs.

6.3 Research Methodology

This study aimed at developing and validating preliminary guideline for creating boundary artefacts in software engineering.

Our motivation for developing such guideline originates from previous research gaps [145] and discussions with a company partner collaborating in this study. We then set the following Research Questions (RQ):

- RQ1: What principles should guide the creation of boundary artefacts?
- RQ2: To what extent can a BA creation guideline be used to create trustworthy boundary artefacts?

- RQ3: To what extent do practitioners perceive the guideline to be effective for creating boundary artefacts?

To answer those questions, we utilised a mixed methods approach performed in three steps (see Fig. 6.1). First, we developed our guideline through a literature review and previous results from our case study [145]. Second, we submitted the guideline for an expert evaluation via two workshops and a survey. At last, we adjusted our guideline by incorporating the feedback obtained during the workshops. We detail the three steps in the following subsections.

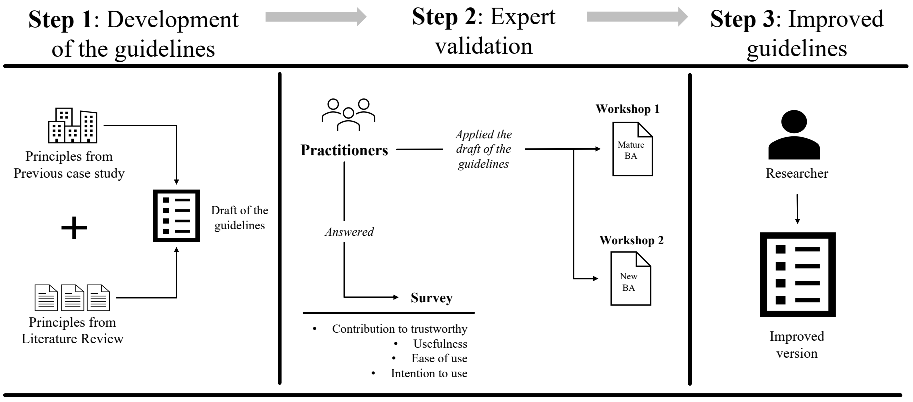

Step 1: Development of the guidelines
Step 2: Expert validation
Step 3: Improved guidelines
Principles from Previous case study
Principles from Literature Review
Draft of the guidelines
Practitioners
Applied the draft of the guidelines
Answered
Workshop 1
Mature BA
Workshop 2
New BA
Survey
Contribution to trustworthy
Usefulness
Ease of use
Intention to use
Researcher
Improved version

Figure 6.1: Research approach followed for developing the guideline.

6.3.1 Step 1 - Development of the guideline

In the first step (see step 1 in Fig.6.1), we collected input from two primary sources to develop the first version of the guideline: a synthesis of the literature and previous case study results [145].

We performed four searches in the Scopus database with distinct search strings (see Table 6.1). The goal was to find relevant studies that reported guideline, frameworks, checklists, and practices utilised to create and manage BAs. We did not limit ourselves to the software engineering literature. We made the search as broad as possible.

Please note that our goal was not to perform a rigorous systematic literature review as widely established by Kitchenham et al. [100]. We selected the studies

Table 6.1: Search strings utilised for finding relevant papers.

Search Strings	Date	Hits	Number of Selected Papers
1 - (("boundary artefact" OR "boundary object") AND (feedback OR "feedback loop"))	10/2022	27	1
2 - ("Designing boundary artefact" OR "designing boundary objects")	10/2022	2	0
3 - ("creating boundary artefact" OR "creating boundary objects")	10/2022	4	2
4 - (("boundary artefact" OR "boundary objects") AND ("software development" OR "software engineering"))	10/2022	65	9

by screening their abstracts and introduction. We provide the list of selected studies and principles collected from each in the supplemental material.

We also added principles extracted from our previous case study [145]. Later, we aggregated similar guides into three categories, detailed in subsection 6.4.2.

6.3.2 Step 2 - Expert validation

In our study, we validated our guideline with experts and collected feedback for further improvement (see step 2 in Fig.6.1). We also collected their perception of the guideline regarding the following rubrics (details of the dynamics of the workshop can be taken from the supplementary material):

- **Appropriateness criteria of the guideline** - how practitioners perceive the guideline to be appropriate for creating BAs.
- **Contribution to trustworthiness** - The degree to which practitioners perceive the guideline to contribute to creating trustworthy BAs. Here, we utilise the trusting beliefs proposed by Lansing and Sunyaev [108]: reliability, Functionality, and Predictability.
- **Ease of use, Usefulness, and Intention to use** – We followed the theoretical model for validating information system design methods [135]. We evaluate the guideline concerning the degree to which practitioners believe the guideline would be: free of effort (ease of use) and the degree to which they believe that the guideline will be effective in achieving their intended objective (usefulness). Lastly, the degree to which practitioners intend to use the proposed guideline.

We performed our workshops at Axis Communications, which provides network solutions in video surveillance, access control, intercom, and audio systems. Axis has over 4,000 employees in over 50 countries, and its headquarters is in Lund, Sweden, where the validation activities were centred.

The workshops lasted around 90 minutes each. A total of six practitioners applied the guideline to two BAs in different maturity stages. In both workshops, we recorded the audio for later collecting their feedback.

Workshop 1

Five practitioners participated in the first workshop. They utilise the BA1 daily, creating or applying its content as input for their tasks. BA1 has been largely utilised in the company for many years. In this workshop, the practitioners examined the BA against our guideline to verify whether they matched the current practices, the status of each principle, and how important they were for the future.

Description of the BA1: It presents the core list of features of all the products in a software product line of the company (covering over 200 products). This content is stored in an XML file with the developed features for the firmware used in the company's products, whether new, modified or deprecated.

Workshop 2

In the second workshop, three practitioners from workshop 1 also joined due to their close connection to BA2 - which was initially planned to complement BA1. In this workshop, practitioners would analyse if the principles would be relevant in creating BA2 and briefly discuss if and how they would implement them.

Description of the BA2: It describes the logical relationship between facts that are expected to be true, e.g. a rule could be "if a product in the software product line has Feature A, then the product shall also have Feature B". The rules are described in YAML format and can be used to automatically detect whether products in the product line follow the rules.

Survey

At the end of the second workshop, the first author sent a survey (see table 6.2) to all participants in both workshops. In that survey, they would answer how they perceived each principle contributing to the trust factors' reliability,

Table 6.2: Survey Questions

RUBRICS	QUESTIONS
Predictability	To what extent do these principles contribute to the content always being available?
Reliability	To what extent do these principles contribute to the correctness of content in an artefact?
Functionality	To what extent do these principles contribute to the proper functionality of the artefact?
Ease of use	Please indicate the degree you believe the guideline are easy to follow.
Usefulness	Please indicate the degree you believe the guideline are useful.
Intention to use	How likely are you intended to use the guideline? Under which conditions would you use it?

predictability and functionality. They also evaluate the guideline' usefulness, ease of use, and intent to use. A total of five practitioners answered the survey. The data analysis from the survey was done through descriptive statistics, focusing on the frequency of the responses. We also utilised stacked bar charts for visualisation (see subsection 6.4.3).

6.3.3 Step 3 - guideline Improvement

We collected feedback on the guideline during both workshops and adjusted its design. The feedback focused on the relevance of each principle to the guideline, general structure, and clarity of the content. After both workshops, we improved our guideline, which we detail in subsection 6.4.2.

Regarding the presentation of the guideline and the placement of the principles, the guideline went through one revision after they were first planned. The first version had four categories. The additional category (shared understanding) had one principle (Define an approach to deal with different interpretations of terminology). However, for simplicity, after receiving feedback from practitioners, we relocated the principle to the category Structure justified by its close connection to the theme. When designing the BA, identifying stakeholders' needs already provides hints on how terminology will differ when crossing borders.

We also moved principle 3d. Agree upon a balanced formality that was initially in the Structure category as it was more coherent with the scope of the category Management.

Changes relating to interpretation and misinterpretation focused on offering better and clear descriptions of the principles. For example, establishing ownership descriptions had too many academic terms, which made it difficult for practitioners to understand. They also asked for standardization of the writing (using active verbs to start) and the definition of each category.

6.3.4 Ethical concerns

Our ethical concerns approach focused on three main aspects: data collection, data analysis, and handling the company's confidential information. We designed and executed our study per the guideline for ethical research provided by the Swedish Research Council [183].

In the data collection phase, we carefully elaborated the questions to guide the workshop not to raise intense emotions or provoke emotional harm [4] (see also our supplementary material). We also disclosed the study's goal and how and when they could remove the data they provided. Also, we explained how the data would be utilised and anonymised through informed consent (see supplementary material).

In the data analysis phase, we anonymised quotes from practitioners and confidential information about the BAs utilised in the workshops. When reporting the results, we carefully paid attention to not stigmatise or harming specific populations. A company employee also checked the handling of confidential information in our manuscript before submission.

6.4 Results

In this section, we report on the results of our study. Besides explaining how we gathered the principles (see subsection 6.4.1), we divided it into two major subsections, which report the latest version of the guideline (RQ1) and the results of the expert validation (RQ2 and RQ3). We also display an example of the application of the guideline at Axis (see subsection 6.4.4).

6.4.1 The organisation of the guideline

The developed categories emerged from our analysis as we collected the principles from the literature. When collecting them, we observed similarities in what they referred to, which made sense in gathering them into categories. For example, principles 2a and 2b (Decide on the format of the artefact and Establish the means of spreading the artefact) strongly connect with how people think about arranging the content, its layout and which channels work best for spreading it. Thus, for principles involving content manipulation, we grouped them into the category structure.

6.4.2 The BA-Guide: Preliminary guideline for creating boundary artefacts in software engineering

The BA-Guide (see Table 6.3) provides ten principles supporting the creation of boundary artefacts in software engineering. We describe the principles and supplement them with examples of implementation. We organise the guideline into scope, structure, and management categories.

1 - Scope

The scope category displays the principles referring to the definition of the target audience in each boundary, their needs regarding the artefact, and how different terminology is held across boundaries.

2 - Structure

The principles in this category relate to how the artefact's content is structured to meet stakeholders' needs regarding format, level of detail, and formality.

3 - Management

The category refers to principles that can guide the establishment of practices to manage the artefact throughout time.

Table 6.3: The BA-Guide: Preliminary guideline for creating boundary artefacts in software engineering

CATEGORY	PRINCIPLES	DESCRIPTION
1. SCOPE	1a. Identify the stakeholders and the boundaries	Identification of the stakeholders that will utilise the artefact's content and their information needs. They affect the usage of the content and how the content is created [37, 145, 207]. Boundaries are usually blurry and often do not relate to a limit or edge but a shared space. They can be teams, departments, units, a group of teams, or, more commonly, organisational domains. It is important to identify them, as the terminology can have several meanings when crossing boundaries. After the workshop with the practitioners, we observed that it is easier to start identifying all the stakeholders that will use the BA and, later, draw the boundaries among them.
	1b. Define an approach to deal with different interpreta-tions of terminology	Refers to how the terminology will be translated through several boundaries [91, 145, 207]. Will the artefact have a glossary of terms or provide commonality and differences in processes across the boundaries? Will it have additional materials that help stakeholders interpret the terminology?
2. STRUCTURE	2a. Decide on the format of the artefact	Refers to how the content is organised and packed in the artefact [207]. The format in which the content is delivered affects how stakeholders will utilise it. Which format is the best? Does it attend to all stakeholders' needs? XML files? Pictures? Which one attends to all stakeholders' needs?
	2b. Establish the means of spreading the artefact	Refers to how the artefact will cross the boundaries and reach the stakeholders [207]. People involved in creating such artefacts should pay attention to how the audience will engage with the content and plan for a simple interaction of finding and utilising the content. As an example of the means, we could consider using Whiteboards, Git repository, Webserver, Confluence pages, etc.
	2c. Decide an appropriate level of content detail	Refers to the level of detail of the content. Enough details in the content are recommended, so the stakeholders find the exact information being searched. If there is too much content, the artefact can be polluted with unnecessary details and decrease its utilisation. On the contrary, the stakeholders will create workarounds that can produce scattered information [201].
	2d. Accommodate additional and experimental content	Refers to the flexibility of the artefact regarding testing new content and increasing its amount. Software development is a dynamic activity that usually welcomes changes while developing and producing experimental content [145, 189]. Does the artefact allow adding additional content (flexibility) to supply any particular need? Thinking about the future, does the actual artefact allow the addition of experimental content that can be incorporated or removed at any time?

Table 6.3: The BA-Guide: Preliminary guideline for creating boundary artefacts in software engineering (continued)

CATEGORY	PRINCIPLES	DESCRIPTION
3. MANAGEMENT	3a. Set up an evaluation and feedback process	Refers to developing a periodic evaluation of the content of the artefact (if it fulfils stakeholders' needs) and a plan for receiving and integrating feedback [37, 91, 145, 207]. The evaluation planning depends on the utilisation of the artefact. Will it be mostly automated? Maybe analysing bug reports or accessing logs of a specific tool can support the evaluation. Questionnaires also work for any artefact. They are simple to create and disseminate. At the same time, you can collect both feedback and an assessment.
	3b. Establish ownership of the artefact	Refers to the distribution of responsibility over the artefact content (or parts of it) and managing changes and improvements [189, 207]. Implementing this guide can reduce incorrect or missing content in the BA and should be determined as soon as possible so that inconsistencies can be found and fixed in the early stages.
	3c. Agree upon a balanced formality	Refers to the level of formality applied to the management of artefacts [207]. How many people are needed to authorise the artefact's incremental changes? How many steps are needed to implement new content/feedback? The level of formality should satisfy stakeholders' needs without being burdensome to create and maintain.
	3d. Handle scattered information	Refers to avoiding and monitoring where and why parts of the content are scattered [201]. The verification can be done during the feedback collection, checking if stakeholders' needs are satisfied. Making the information easy to find and collect minimises the risk of stakeholders forking the content to create other artefacts that suits them better.

6.4.3 Contribution of the guideline to creating trustworthy artefacts

After conducting the workshops, we sent the survey to collect the practitioners' perceptions on how the guideline' principles contributed to creating trustworthy BAs (RQ2). We display the results in Fig. 6.2. We also show the practitioners' evaluation regarding the ease of use, usefulness and intention to use the guideline.

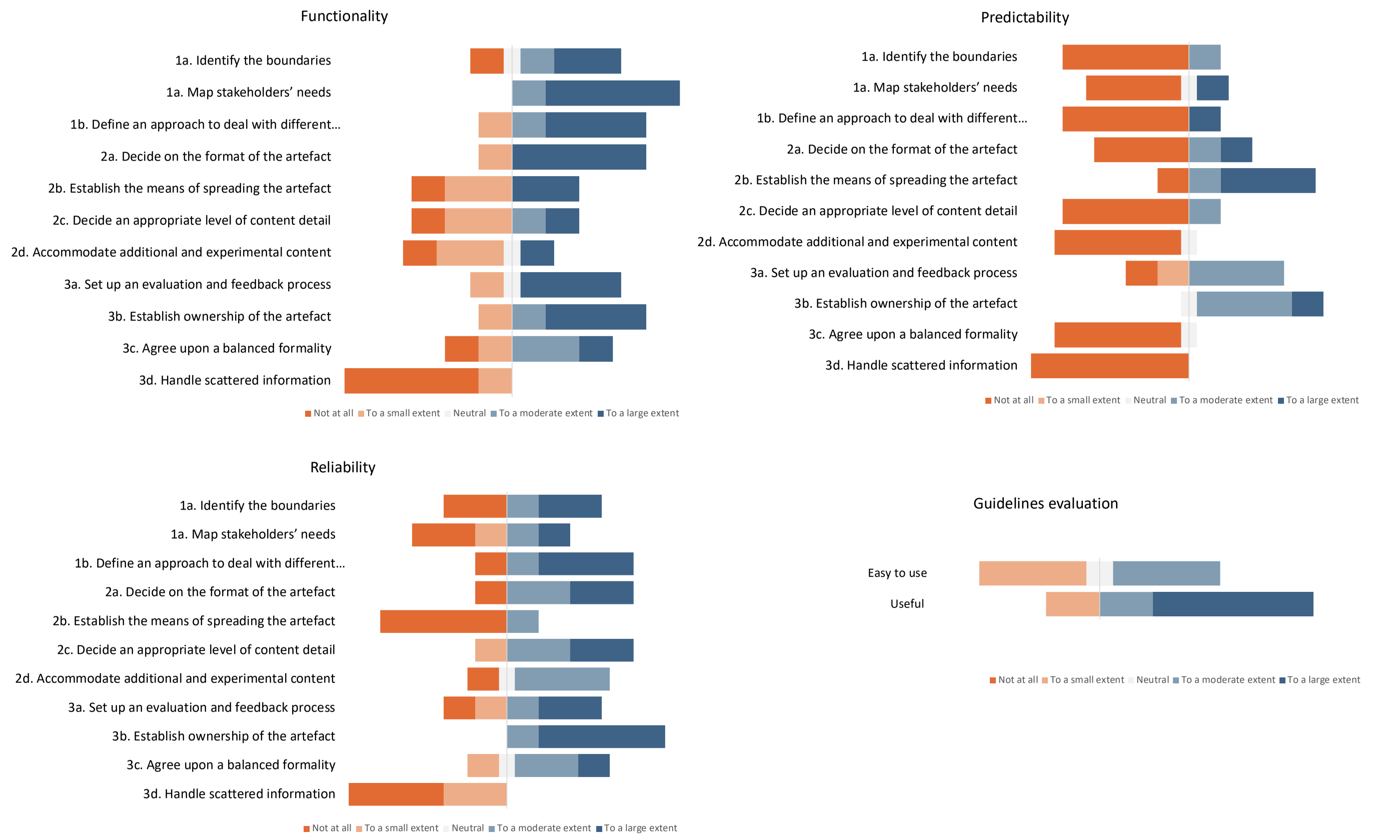


Figure 6.2: Contribution of the guideline to trust factors

We asked the practitioners how they perceived the guideline contributing to the BA performing as needed for the task environment (**Functionality**). In Figure 6.2, the practitioners perceive that seven principles contribute positively to functionality. The functionality is achieved mostly by the principles that outline the BA (category Scope) and establish practices to manage and keep (category Management) its construction.

When taking a closer look at principles 1a, 2a, and 1b, we observe that stakeholders expect that the content matches their needs and is displayed in a format that facilitates its usage. As the BA crosses boundaries, its functionality is enhanced by translating terminology to the different stakeholders.

The **Reliability** results are similar to the functionality, with a small, still positive, variation between moderate and large extent. However, principle 2b - Establish the means of spreading the artefact does not seem to contribute to the artefact providing accurate content, as pointed out by the practitioners. One reason for this result is that spreading the content occurs one step after the content is prepared (created or fixed). Therefore, it does not directly affect the accuracy of the content.

On the other hand, the practitioners consider principle 2b relevant for the artefact content is always provisioned as requested (**Predictability**). The other two principles are positively influencing the predictability of BAs. They are the existence of practices for evaluation and feedback implementation and the establishment of ownership, which can guarantee that the BA is continually managed.

The practitioners, on the whole, consider principle 3b - Establish ownership of the artefact as a positive and relevant principle that can contribute to the artefact being predictable, reliable and offering proper functionality.

In contrast to the other principles, principle 3c - handle scattered information, was not seen as contributing to any of the trust factors analysed during the workshops. During the discussions in the first workshop, the practitioners understood that if principle 1a (Map stakeholders' needs) is well-taken care of, scattered information won't probably happen. In the second workshop, they did not consider this principle when creating a BA simply because it did not exist yet, so scattered information was nonexistent.

In the final part of the survey, practitioners were asked about the guideline' ease of use, usefulness, and intention to use them. Figure 6.2 displays the results. There is no consensus among the practitioners that the guideline are easy to use. When crosschecking these results with the feedback received during both workshops, we believe that the practitioners had a hard time understanding the concept of BA and the complexity of the written language we utilised.

By contrast, most practitioners believe that the guideline are useful to a large extent.

Concerning the intention to use, practitioners manifested a positive intention to apply the guideline in both artefacts that we utilised in the workshops. As one of the practitioners pointed out, "*one benefit of having some guideline like this is that if many boundary artefacts follow the same guideline or template, that will also make it easier for people to consume the boundary artefact. That makes me more convinced that we should use this even if it is not 100% perfect for everything; it will help people to align some of the different kinds of boundary artefacts we have and help people to work with them.*"

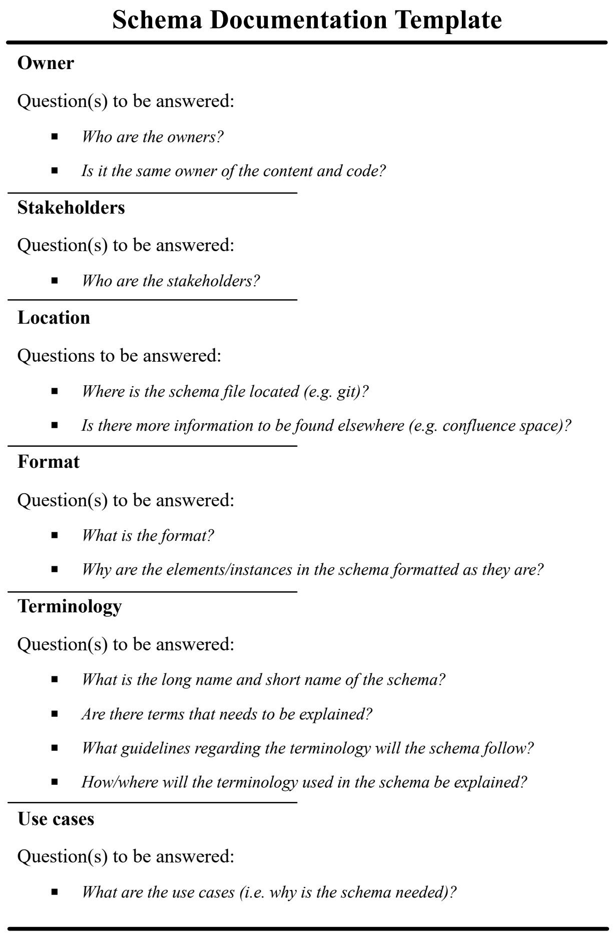
Together these results provide important insights into the creation of trustworthy BAs. As the way the BA is structured and managed can influence stakeholders' behaviour, it is crucial to understand how we can drive this behaviour towards a positive attitude on which they can rely on them [145]. On the contrary, those knowledge resources will be underused and potentially increase the development costs due to the creation of complementary artefacts or wasting time trying to find the correct information.

6.4.4 Example of application of the guideline

After our last workshop, the practitioners developed a preliminary version of a template to guide schemas documentation creation (see Figure 6.3). The practitioners use schemas (i.e. common definitions of structured data in computer-readable format) to facilitate the automation of information flow.

A schema that has been used by the practitioners (BA1) had originally been poorly documented, and the hope is that better documentation, especially explicitly stating the Stakeholders, Terminology and Use cases, would help new schemas (such as BA2) to avoid some of the trust issues that were experienced with BA1. The schema document template was created to help creators of new schemas identify the minimum set of questions to be answered and documented for new schemas and align how different schemas are documented.

BA2's immediate stakeholders are people within the QA department, other R&D departments, and Product management. The practitioners also foresee that there will be interest in BA2 outside these departments, e.g. in Sales or Marketing.



Schema Documentation Template

Owner

Question(s) to be answered:

- *Who are the owners?*
- *Is it the same owner of the content and code?*

Stakeholders

Question(s) to be answered:

- *Who are the stakeholders?*

Location

Questions to be answered:

- *Where is the schema file located (e.g. git)?*
- *Is there more information to be found elsewhere (e.g. confluence space)?*

Format

Question(s) to be answered:

- *What is the format?*
- *Why are the elements/instances in the schema formatted as they are?*

Terminology

Question(s) to be answered:

- *What is the long name and short name of the schema?*
- *Are there terms that needs to be explained?*
- *What guidelines regarding the terminology will the schema follow?*
- *How/where will the terminology used in the schema be explained?*

Use cases

Question(s) to be answered:

- *What are the use cases (i.e. why is the schema needed)?*

Figure 6.3: Example - Schema Documentation Template

6.5 Discussion

We followed a multi-step approach to develop and validate our preliminary guideline in an industry context. Our work expands on previous research on BAs by providing structured guideline gathered into three main categories (scope, structure, and management) that help practitioners create BAs in software engineering contexts. As the BA literature in software engineering contexts is scarce, this guideline is significant and unique because they offer an original perspective on essential principles that practitioners should observe upon the conception of a BA.

Further, in our guideline, we incorporated extendable practices applied to specific contexts such as product family development and the automotive industry [91, 201]. These were considered relevant for the context we utilised in this study, which we believe has a potential for generalisation of the principles as more applications of the guideline may occur.

The current version of our guideline evolved based on practitioners' feedback after conducting two workshops. Besides rearranging principles into categories (see subsection 6.3.3), the latest version differs from the first to meet practitioners' needs for clarity in describing each principle, especially regarding the concept of a boundary. During the workshops, they agreed with the principles' importance, although most were not implemented.

However, the concept of the boundary was difficult to understand, even when having a well-established BA. Attending to stakeholders' needs was tacitly made, but not all found the needed content. As we observed this event, we realized that understanding this concept is the first step in providing *shared understanding*, which is one critical aspect of a well-functioning BA [175].

Practitioners considered the **category scope** quite logical. However, they did not have practices targeting stakeholders' needs and terminology. They also seem unfamiliar with how the boundaries can affect the BA. When not planned or managed, artefacts tend to grow organically. As people start to use them, BAs cross several boundaries and become overloaded with multiple meanings, losing their value and decreasing trust levels [81,145,175,206]. In this situation, stakeholders are not well known, which makes their needs neglected. The scope category targets those issues and directs what to consider when creating BAs, especially determining a satisfactory level of *plasticity* of the BA, which attends to stakeholders' local needs and still keeps its robustness.

The **structure category** brings new insights into principles particular to software development contexts, such as accommodating additional and experimental content [145,189]. Practitioners already acknowledge some of the principles belonging to the structure category, such as format and means of spreading, and how considering them enhances the *ability to transfer* information among different social worlds. However, they are unfamiliar with all stakeholders' needs, focusing primarily on immediate users. Another important remark of this category is the inclusion of the principle "decide an appropriate level of content detail", which, according to the practitioners participating in this study, can reduce the chances of practitioners forking out content and creating their own artefacts.

The **management category** had a surprising and positive reaction from practitioners in both the survey and workshop, especially regarding the principle of "establish ownership of the artefact". Ownership has been explored in software engineering literature but primarily focuses on code [63, 127, 142, 204]. We emphasise that this principle, aligned with evaluation and feedback, shapes a structured strategy for managing BAs and keeping their consistency [175]. Besides, their implementation creates a favourable environment for stakeholders to feel confident about relying on BAs, as they trust the content is correct and updated.

Artefacts, in general, are recognized as necessary, but still, practitioners see them as burdensome to create and maintain [177]. Trust decreases when the produced artefact is neglected, and its use is highly affected. The results of our study have shown that practitioners perceive that the guideline contribute to creating trustworthy BAs through the three favourable factors, reliability, functionality, and predictability.

The contribution, though, differs between factors. Principles in the categories of Scope and Management strongly contribute to the BA being functional and reliable. Whilst the predictability factor has strong support from fewer and dispersed principles through the three categories, such as establishing ownership and establishing the means of spreading. These results are elucidative because, as a whole, the principles support the creation of trustworthy BAs. However, the principles can be utilized individually in specific occasions where companies, for example, want to fix a singular reliability issue and need to know which principles increase trust in BA, observing how they contribute the most to reliability.

One important note is worth mentioning. The guideline should not be seen as a process which requires all principles to be implemented in the cited order. But rather, the principles should be tailored to each BA and its context, meaning they will have different levels of importance and implemented practices.

These findings, while providing an important contribution to the software engineering literature with guideline that were inexistent by the period in which this research was concluded, also leave open questions. Would these principles remain significant in other contexts with different BA formats? Would the number of principles increase, decrease or remain the same?

We consider the developed guideline preliminary; we expect them to evolve and reveal differences as more applications are executed. Thus, new principles may be added to the guideline to match different contexts and BA, which can grow evidence in this study area.

Following our first questions, the trust factors can also differ in their contribution to trust in general. But how do they differ? Would the principles 3d. Handle scattered information, for example, have different results? We suspect that yes. Therefore, other studies could focus on identifying differences in the principles' contribution to trust when the type of BA change. The results could provide interesting insights into which principles are decisive in keeping the different BAs trustworthy.

6.6 Lessons learned

Overall, our approach was well-received by our company partner, with some initial difficulties in understanding key concepts. The most important lessons learned for us are the following:

Lesson #1: Planning BAs is not an isolated activity.

The design of the BA is a group activity that involves people of different social worlds. As we gathered people that sometimes belonged within the same boundary, their different views helped to address the difficulties of what and how to represent information in the BA. Additionally, this production process allowed the solution of potentially conflicting interests [175].

Lesson #2: The process of creating BAs is time-consuming.

Thinking about the structure and reconciling different perspectives requires much more than a single meeting. Usually, other stakeholders need to be involved until a consensus is reached. Initial gathering focuses mostly on brainstorming sessions where people would read the guideline and think of how to implement such principles into the artefact. As the discussions progress, the practices start to be established.

Lesson #3: Start with a few boundaries and increase them as there is a need for the artefact to grow.

At the beginning of a BA creation, sometimes the boundaries do not appear straightforward, and even when they are, dealing with all of them can be complex. During the workshops, we learned that visualising how potential boundaries, e.g., one or two, could utilise already shaped directions on implementing the guideline.

> **Lesson #4**: Reviewing the stakeholders' information sources for performing tasks can facilitate outlining their needs.

Knowing the stakeholders' existing sources of information can help define the level of detail of the information added to the BA. In our workshop with BA1, some practitioners were unsure about keeping one specific piece of information as it had no owner and was updated. However, during the discussions, they found that the stakeholders could not find that information anywhere else. Therefore, they decided to keep and appoint an owner for that content and keep it up to date.

> **Lesson #5**: Mapping the guideline against a new (or existing) BA reveal preconceived assumptions.

During our workshop with BA1, practitioners realised that many assumptions were made while creating the artefact. When they confronted the few existing practices with the guideline, they exposed the lack of decisions on important properties of the BA, such as management of the BA, approach for different terminology and stakeholders' needs.

6.7 Threats to validity

In this section, we report and discuss threats to the validity of our study. We group the threat according to the classification proposed by Wohlin et al. [200]: construct, internal, external, and conclusion threats.

Construct validity refers to threats to the interpretation of theoretical constructs investigated and how they were actually investigated in the research setting. We identified one threat to practitioners' understanding of the guideline and trust factors in both workshops and surveys. During the workshop, we clarified misunderstandings and definitions of complex constructs such as boundary artefacts to minimise this threat. Besides, we avoided using the trust factors to avoid confusion or misinterpretation. Instead, we utilise questions that are straightforward to understand (see supplemental material).

Internal validity concerns the awareness of other factors that can affect the outcome of the findings other than the ones already identified. We acknowledge the existence of this threat when collecting the principles and creating the guideline. To minimise this threat, we made sure not to exclude any domains to synthesise state-of-the-art practices when creating and managing BAs.

External validity relates to the extent that the results are generalisable. To increase the generalisability of the guideline, we adopted knowledge from different areas and did not limit ourselves to the software engineering context. Yet, we acknowledge the number of participants and the type of the artefact (mostly automated) could not be enough for a significant generalisation factor. However, generalisation was not our goal; rather, it was to provide preliminary guideline that can be gradually adopted and extended. Hence, as more applications of the guideline occur, we expect the guideline to change over time to accommodate characteristics of different artefacts or contexts within software engineering. In addition, differences in practitioners' perceptions of how the principles contribute to trustworthy artefacts in other contexts.

The conclusion validity refers to how consistent the data collection allows for drawing accurate conclusions. To synthesise the literature, we applied some structure to find relevant papers. We also recorded the workshop not to miss the feedback and discussions over the principles. At last, the second and third authors joined discussions regarding designing the guideline, the structure of the workshops, and the results collected afterwards.

6.8 Conclusion

Boundary artefacts are critical for software development because they can cross several boundaries in the organisation and provide relevant input to different stakeholders. Yet, they are doomed to failure if they have enough inconsistencies that drive stakeholders not to use them, negatively affecting software projects.

In our investigation, we developed preliminary guideline that advise creating BAs. In addition, through an expert validation of the guideline, we collected practitioners' perceptions on how each guideline's principles contribute to creating trustworthy BAs. We grouped the principles collected from a literature review into three categories. The first category (Scope) focuses on the scope, displaying principles referring to defining each boundary's target audience, needs, and terminology. The second category (Structure) relates to how the artefact's content is structured to meet stakeholders' needs. The third (Management) refers to principles that can guide the establishment of practices to manage the artefact throughout time.

The expert validation revealed that the principles contribute to creating trustworthy BAs at different levels—also, the relevance of the guideline and their usefulness. Moreover, the guideline strengthen BA traits such as shared understanding, plasticity and ability to transfer. Practitioners can utilise the guideline to guide the creation or even evaluate current practices for existing BAs.

As the principles contribute differently to trustworthiness in our specific case, we believe that future application of the guideline and trust evaluation with different types of BAs, i.e. manual or automated, could provide interesting results on how the principles contribute to trust depending on the type of BA.

Chapter 7

Connecting the Dots of Knowledge in Agile Software Development

This chapter is based on the following paper:

> Raquel Ouriques, Tony Gorschek, Daniel Mendez, Fabian Fagerholm. "Connecting the Dots of Knowledge in Agile Software Development". Submitted to *Communications of the ACM (CACM)*(Under review), (2023).

7.1 Introduction

Knowledge is the fundamental resource for executing software development activities [161], put under the spotlight when agile principles started to be incorporated by companies. The shift from traditional methods, which included extensive planning and heavy documentation, now emphasises flexibility and focus on people and their knowledge to deal with requirements, design, and implementation evolving iteratively [60].

Collaboration and communication became the critical facilitator for people engaging in creative gatherings for knowledge sharing within self-organised teams [39]. However, knowledge, as an intangible resource, is difficult to reproduce and manage, requiring significant effort to understand in terms of what should remain tacit, and what should be explicitly captured in artefacts, like documentation.

The importance of knowledge as a resource rests on its great potential to create economic value when combined with other people's knowledge, which in turn provides the means for companies to capitalise on it. In this article, we'll talk about why it is so challenging to manage knowledge, understand the proportions of knowledge that remain intangible, and provide insights into how we can effectively manage knowledge stored in artefacts. This is complicated by these key factors:

1. Knowledge belongs to people.
2. Knowing what knowledge should be added to artefacts is complex.
3. Managing knowledge in artefacts requires more coordination than is obvious.

We will also discuss types of knowledge, types of artefacts, and what to add to artefacts. This work is based on studies carried out in industry and reported in detail in peer-reviewed journals [145, 147, 148].

7.2 Knowledge Resources

Before diving into knowledge waters, we first reflect on companies' competitive resources and how they relate to knowledge. Optimisation and efficient allocation of resources are concepts that historically relate to the manufacturing industry. Resource management primarily focuses on planning, allocating people, money, and technology to maximise organisational value [11].

This way of thinking is still valid in the development of software-intensive products and services. However, the proportions of the type of resources have changed. In the software industry, co-workers are the critical and most important resource, as their knowledge can generate sustained competitive advantages for companies.

An intangible portion of co-workers' knowledge, which we refer to as **Knowledge-based Resources** (KBRs) (see illustration in Figure 7.1), is revealed when stored and applied through, e.g., prototypes, architectural descriptions, requirements, and other artefacts. An artifact refers to a tangible or intangible product generated, modified, or utilized in a series of tasks that hold value for a particular role [132]. As this knowledge is typically stored in artefacts (documents) or

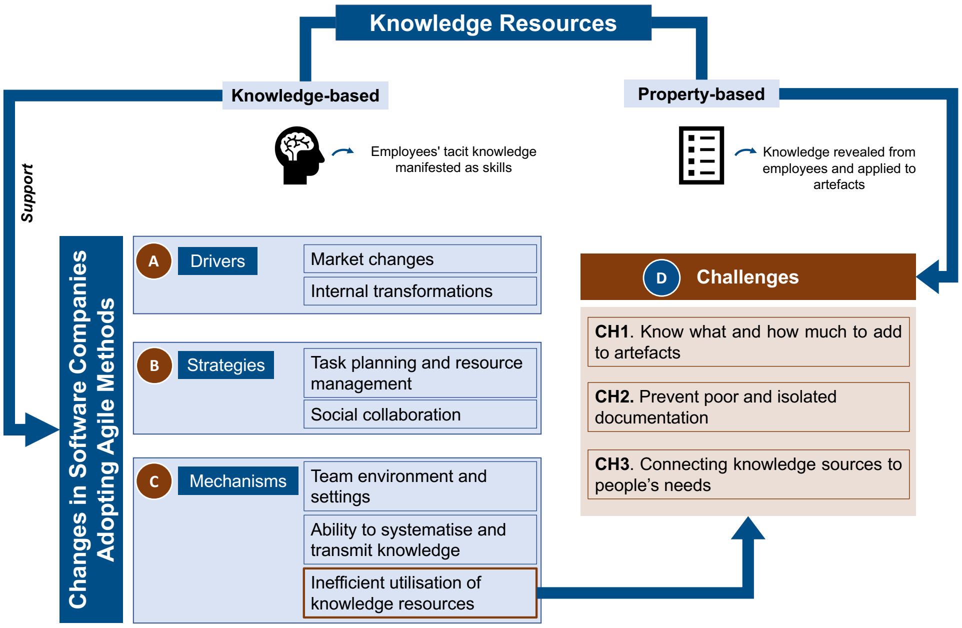
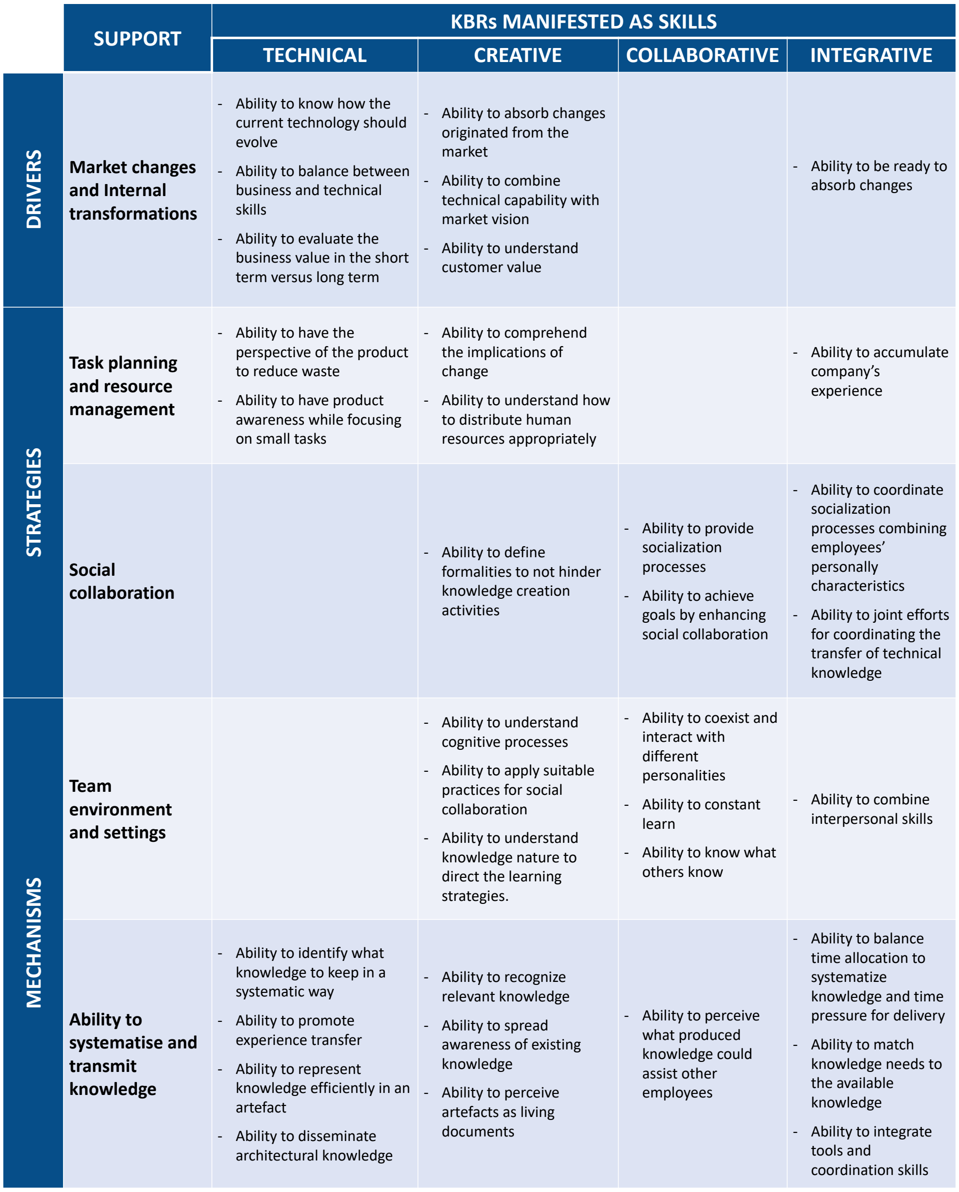

Knowledge Resources
Knowledge-based
Property-based
Employees' tacit knowledge manifested as skills
Knowledge revealed from employees and applied to artefacts
Support
Changes in Software Companies Adopting Agile Methods
A Drivers
Market changes
Internal transformations
B Strategies
Task planning and resource management
Social collaboration
C Mechanisms
Team environment and settings
Ability to systematise and transmit knowledge
Inefficient utilisation of knowledge resources
D Challenges
CH1. Know what and how much to add to artefacts
CH2. Prevent poor and isolated documentation
CH3. Connecting knowledge sources to people's needs

Figure 7.1: The types of knowledge resources. The figure shows how knowledge-based resources support changes in agile and the challenges involved in their utilisation to create property-based resources [147].

transformed into products, they become **Property-based Resources** (PBRs) as companies own them and can generate profit utilising them [134].

However, a large proportion of these KBRs remain intangible, and are, thus, not realised into PBRs. There might be several reasons for this. The domain of the company might be moving too fast to create PBRs and a PBRs might become outdated too fast to warrant the formalisation investment. Another possible scenario is the complexity of generating PBRs out of KBRs that are usable and useful for others. Knowledge in terms of KBRs are thus a reality that companies live with, and they strongly support software development and enable companies.



	SUPPORT	KBRs MANIFESTED AS SKILLS			
		TECHNICAL	CREATIVE	COLLABORATIVE	INTEGRATIVE
DRIVERS	**Market changes and Internal transformations**	- Ability to know how the current technology should evolve - Ability to balance between business and technical skills - Ability to evaluate the business value in the short term versus long term	- Ability to absorb changes originated from the market - Ability to combine technical capability with market vision - Ability to understand customer value		- Ability to be ready to absorb changes
STRATEGIES	**Task planning and resource management**	- Ability to have the perspective of the product to reduce waste - Ability to have product awareness while focusing on small tasks	- Ability to comprehend the implications of change - Ability to understand how to distribute human resources appropriately		- Ability to accumulate company's experience
STRATEGIES	**Social collaboration**		- Ability to define formalities to not hinder knowledge creation activities	- Ability to provide socialization processes - Ability to achieve goals by enhancing social collaboration	- Ability to coordinate socialization processes combining employees' personally characteristics - Ability to joint efforts for coordinating the transfer of technical knowledge
MECHANISMS	**Team environment and settings**		- Ability to understand cognitive processes - Ability to apply suitable practices for social collaboration - Ability to understand knowledge nature to direct the learning strategies.	- Ability to coexist and interact with different personalities - Ability to constant learn - Ability to know what others know	- Ability to combine interpersonal skills
MECHANISMS	**Ability to systematise and transmit knowledge**	- Ability to identify what knowledge to keep in a systematic way - Ability to promote experience transfer - Ability to represent knowledge efficiently in an artefact - Ability to disseminate architectural knowledge	- Ability to recognize relevant knowledge - Ability to spread awareness of existing knowledge - Ability to perceive artefacts as living documents	- Ability to perceive what produced knowledge could assist other employees	- Ability to balance time allocation to systematize knowledge and time pressure for delivery - Ability to match knowledge needs to the available knowledge - Ability to integrate tools and coordination skills

Figure 7.2: Knowledge-based resources manifested in skills mapped to the drivers, strategies and mechanisms to adapt to changes in agile software development [147].

7.3 From Artefacts to People

Integrating agile principles into software development altered the function of artifacts. The emphasis shifted towards intense communication, offering increased flexibility in adjusting priorities and reallocating workloads.

Knowledge-based resources are considered essential in this environment as they can contribute significantly to innovation and companies' performance. *KBRs often manifest as specific skills, including technical, creative, integrative and collaborative skills* [134], which we describe and map to the drives, strategies and mechanisms to deal with changes in agile contexts in Figure 7.2.

The importance of these resources and focus vary depending on the industry domain [5]. Even software companies will have different levels of priority. However, something they all have in common is constant changes. So, what role do KBRs play when companies need to adapt to change in agile contexts?

First, we must understand where these **changes** come from. They are **driven** by market changes or internal transformation (see A in Figure 7.1). Market changes refer to all external events that force companies to adapt, i.e., technological breakthroughs, new business opportunities, and customer demands. Internal transformations refer to internal decisions that induce organisational changes, i.e., technological adaptations and resource rearrangements. In these scenarios, co-workers utilise their skills and technical capabilities to, for example, analyse the impact of changes on current products, also foreseeing how the product should evolve to match future technologies.

Changes can also spark internally, i.e., re-organisation for efficiency, agile transformation, and product innovation processes. These changes will trigger KBRs, which play a strategic role in incorporating and adapting to changes. Herewith, we can say that KBRs:

1. Help examine internal and external variables affecting the company and define how to respond and incorporate potential changes.

Second, as the changes start to be assimilated, companies must create **strategies** (see B in Figure 7.1) to disseminate new arrangements originating from changes. They drive co-workers to achieve particular goals through social collaboration, combining their knowledge to deal with the complexity of the products and coordinate actions. For example, by having the technical skill of product awareness (commonly known as big picture), it is possible to evaluate how a new requirement affects the product and its ramification through product development. As it stands, KBRs support the implementation of strategies to deal with changes by:

2. Enhancing social relations within agile teams, encouraging knowledge sharing to solve complex problems associated with software development.

3. Perceiving the changes and their ramifications through product development and adequately adjusting tasks and resources.

Software companies also utilise different **mechanisms** (see C in Figure 7.1) to ensure that changes are fully implemented. Effective agile teams is one such mechanism, but to achieve change, companies must understand the key aspects of creating a favourable environment where people feel comfortable collaborating. When this mechanism is established, the ability to store and transmit part of the knowledge shared intuitively becomes crucial for companies to *use KBRs to produce PBRs*. By doing this, companies can hold ownership of the co-workers' knowledge as it is stored in artefacts protected by property rights. KBRs support these mechanisms to:

4. Understand what characteristics are relevant in setting up effective agile teams.

5. Support identifying what knowledge needs to be transformed into PBRs.

It is important to note that even though intuitive actions towards the utilisation of KBRs can lead to successful implementations of changes, a lack of insight into managing knowledge produces **inefficiencies**. The lack of a structured process for capturing and storing relevant knowledge results in an overload of information in the artefacts that constitute the PBRs. In these circumstances, co-workers can carry out meaningless searches, confusion about the location of accurate content, and waste time parsing large artifacts. At last, they can get frustrated because of recurring problems that could have been solved if the information had been concise, timely and correct.

The inefficiencies challenge capturing knowledge and transforming it into PBRs (see D in Figure 7.1). Although the main focus of software companies adopting agile principles is on people rather than artefacts, the proportion of these artefacts' utilised remains critical for the efficiency of the software development life cycle and the principle and practices associated.

7.4 Boundaries are Critical

PBRs can be powerful. A single artefact can carry relevant knowledge through different parts (boundaries) of a company. Doing so, they can provide different meanings. For example, a use case that was initially created during requirements engineering can be utilised in planning, design, estimation, and testing activities. In each of those boundaries, people see the use case from their perspective and execute tasks that ultimately contribute to the same larger goal.

The contribution of such artefacts is undoubtedly large. They are flexible in supplying local needs (as per the use case example in Figure 7.3) while keeping themselves highly structured. Co-workers look at the same information, but what it means to them can differ and usually changes over time. They are known as Boundary Artefacts (BA), and we explore them in this work as they represent a specialisation of PBRs.

This type of artefact provides a critical contribution to companies adopting agile methods. It is a resource when informal communication is not possible. In geographically distributed teams, boundary artefacts are even more important as they can overcome communication barriers and different time zones. Most recently, the pandemic has forced a shift from co-located communication to online formats, bringing many challenges to software teams, including the increased dependence on artefacts.

In some cases, for several reasons, boundary artefacts may not be available or provide accurate content. Co-workers' trust can be affected when such inconsistencies occur, and they become skeptical of using the artefacts [145]. They can avoid using an artefact by searching for information elsewhere, which does not guarantee accuracy either. They can create workarounds to supply that knowledge need, i.e., creating their own versions. The latter becomes dangerous as duplicates can grow, producing uncertainty and even more distrust. Consequences to the software engineering effort can be diverse in such cases and can result in the execution of wrong tasks, utilisation of incorrect information, and setbacks in the development process (see implications in Figure 7.3).

Adjusting the boundary artefact to meet co-workers' needs also has challenges (see challenges described in Figure 7.3) that involve keeping the accuracy of the content and the boundary artefact aligned to its original characteristics. One crucial aspect that has been reported as causing misunderstandings is terminology [91]. When co-workers look at the artefact and do not recognise the terminology, there is a risk of misinterpretation and, consequently, misuse. Even worse is if there is an assumption of a common understanding. In such cases, the benefits of boundary artefacts is undermined. Accuracy and control are also identified challenges that require significant effort to avoid the inefficiency of boundary artefacts.

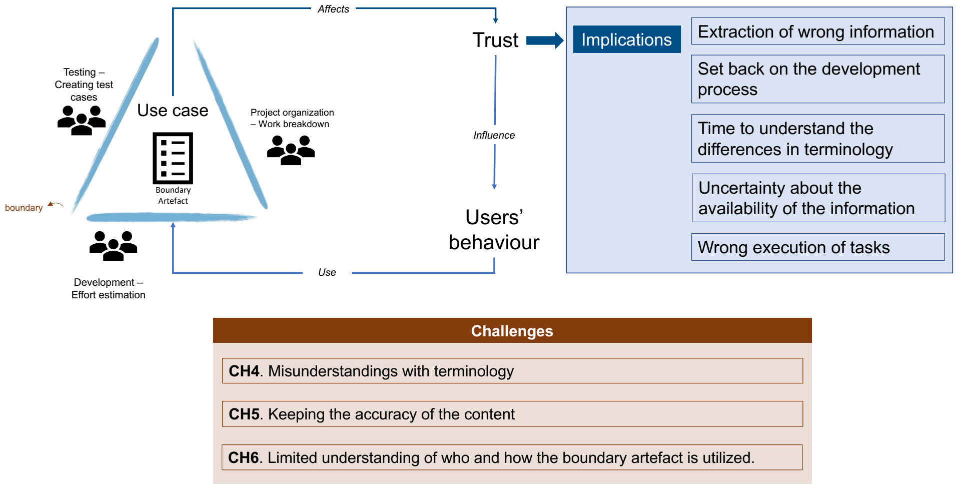

Affects
Trust
Implications
Extraction of wrong information
Set back on the development process
Time to understand the differences in terminology
Uncertainty about the availability of the information
Wrong execution of tasks
Influence
Users' behaviour
Use
Testing – Creating test cases
Use case
Boundary Artefact
Project organization – Work breakdown
boundary
Development – Effort estimation
Challenges
CH4. Misunderstandings with terminology
CH5. Keeping the accuracy of the content
CH6. Limited understanding of who and how the boundary artefact is utilized.

Figure 7.3: The cycle of trust in boundary artefacts and how it influences users' behaviour. The figure displays the negative implications of differences in trust and the challenges to managing these artefacts [145].

7.5 Strengthening Property-based Resources

The key to optimising PBRs is to establish formal and structured practices. Even though artefacts receive less attention in agile environments [?], PBRs play a critical role in companies, addressing the challenges involving utilising KBRs to produce PBRs and boundary artefacts necessary. Here we provide potential solutions originated from empirical work [145, 147] targeting the challenges (see Figures 7.1 and 7.3) revealed in this article.

This contribution originates from two main research results reported in two papers. One case study investigated the causes and effects of trust in boundary artefacts. The second is a grounded theory study to identify KBRs and explain how they supported changes in agile contexts.

7.5.1 Challenges for producing PBRs

CH1. Know what and how much to add to artefacts. When a structured process for adding content to artefacts is missing, co-workers will fill them according to their preference of what they consider good enough regarding content. The content overload will induce co-workers to meaningless searches where they won't find what they need and lack precision, causing confusion. Conversely, searches won't provide enough knowledge required from co-workers. They will start looking for complementary artefacts to fulfil their needs.

- Solution - CH1. The transformation of the co-workers' knowledge should satisfy their knowledge needs [201]. Identifying co-workers' knowledge needs can be time-consuming, though. However, by being precise and structured, the correct amount of knowledge stored as PBRs tends to be more efficient.

CH2. Prevent poor and isolated documentation. One of the consequences of a lack of formal practices is that co-workers can create artefacts within teams without knowing who would benefit from them. Usually, this process is sub-optimal as co-workers are targeting specific internal issues and probably won't maintain that content. In this circumstance, this this practice can overload tools such as Wikis with too much and ill-suited content, which will make it harder, and sometimes confusing, for others to find relevant content.

- Solution - CH2. The creation of official artefacts should be guided by structured procedures that provide at least scope, targeted audience and overall structure [145]. The way the content is structured affects usability and applicability. To help address these concerns, co-workers could think about why people need certain knowledge and how it could be applied. It is worth mentioning that creating such structures does not mean blocking co-workers from creating their artefacts within their teams, bringing rigidity, which goes against agile flexibility. We rather mean that the production of a PBR, which is an investment, should follow an official artefact decision to spread and maintain it.

CH3. Connecting knowledge sources to people's needs. Not all PBRs exist in electronic versions such as PDFs or Wiki-based tools. They could, for example, exist in the form of maps, pictures, or sticky notes. It is important to know the format of the planned PBR, and to develop the best way to "connect" it to co-workers who need it.

- Solution - CH3. In software development contexts, most PBRs remain in electronic format, highlighting the relevance of Information and Communication Technologies (ICTs) for connecting co-workers to relevant sources of knowledge [147, 207]. Software companies largely utilise them for coordination, communication, and storing PBRs. However, having ICTs does not guarantee the applicability of the artefacts stored. One should note that, for PBRs, it only means that many co-workers can access them in several parts of the organisation and different locations. The effectiveness, though, relies on the other aspects mentioned above, such as identifying knowledge needs and making sure that each artefact has a deliberate structure and content.

As we narrow down to specifically boundary artefacts the challenges pertain to the creation and maintenance of boundary artefacts:

CH4. Misunderstandings with terminology. Terminology can confuse co-workers if not properly addressed when creating boundary artefacts. Over each boundary, the content can have different meanings and interpretations; if the terms are poorly defined, co-workers can misuse the content.

- Solution CH4. Knowledge can have different levels of novelty, influencing how terminology will be displayed. Carlile [33] introduces three approaches to dealing with terminology. The first refers to generating a common lexicon or taxonomies to address the differences when having a syntactic boundary artefact. The second suggests using cross-functional interactions and boundary translators when addressing semantic boundaries. Finally, pragmatic boundaries focus on combining different knowledge and for that, utilising artefacts that can be jointly created can help, i.e., prototyping.

CH5. Keeping the accuracy of the content. When the content is unreliable, co-workers will change their behaviour towards a boundary artefact, creating workarounds, not using them, or making personal "versions". This can happen for many reasons, including a lack of control over the content, ownership issues, and a disconnect between users and creators.

- Solution CH5. To deal with the flexibility of agile environments, a cooperative approach is preferred. Ownership of the boundary artefact can be distributed among users/creators, reducing the workload [189, 207]. For example, in technical artefacts, there could be an owner for the technical part, and another for managing the content. Depending on the size of the artefact and diversity regarding boundaries, ownership can be distributed even more. We also suggest implementing periodic evaluation and feedback processes to include improvements to the process and artifacts as suggested by users, maybe via the retrospective mechanism.

CH6. Limited understanding of who and how the boundary artefact is utilised. A key aspect of boundary artefacts is that the content provided in the artifacts reaches predefined users over boundaries. The matching is likely to happen when there have been negotiations and agreements between the different areas of the organization.

- Solution CH6. The simplest solution is to map users' needs, make creators aware, and conduct negotiations [207]. However, this requires maturity in companies when handling boundary artefacts, which is not a guarantee. Most boundary PBRs are well-established artefacts that have been utilized for years despite being inefficient. In this context, we suggest that a diagnosis is made to check if users' content needs are satisfied and map the boundaries for checking terminology differences. As this process requires an investment, it can be more efficient to focus on the severity of negative implications and make the changes gradually focusing on critical PBRs first.

7.6 Reconciling with Traditional Planning

Managing knowledge through PBRs requires more planning and structure than normally associated with agile software development contexts. However, we need to think about the changes software companies have endured since the wide adoption of agile principles, as well as all the principles and practices introduced since then. First, companies have grown and spread development activities across countries. Artefacts have become a key to enabling communication and coordination, as well as being a critical resource for executing tasks.

Second, we have seen lots of evidence showing how the pandemic has changed the ways of working, such as hybrid or completely remote work [170]. These changes challenge face-to-face communication and knowledge sharing. As people search for relevant content stored in artefacts, the need to manage knowledge accuracy increases [145].

There has been a dichotomy between agile adoption (albeit not formally expressed in agile principles) and traditional plan-driven organizations. Combining the best of both views seems reasonable. Dybå and Dingsøyr [60] suggest to exploit traditional planning principles in some cases, especially for larger efforts.

Similarly, including more formal planning in managing PBRs does not mean moving back to a documentation-centred approach, but rather managing the creation and maintenance of reliable, predictable and functional artefacts. The main premise of both lean and agile is to "remove waste" and create value. However, overhead, such as coordination and communication through PBRs, is a necessity to be able to create value. Thus, knowledge artefacts that support value creation should be deliberately handled as part of the development process to maximise their utility.

7.7 Acknowledgments

We would like to acknowledge that this work was supported by the KKS foundation through the S.E.R.T. Research Profile at Blekinge Institute of Technology.

References

[1] Aghaei Chadegani, A., Salehi, H., Yunus, M., Farhadi, H., Fooladi, M., Farhadi, M., and Ale Ebrahim, N. A comparison between two main academic literature collections: Web of science and scopus databases. *Asian Social Science 9*, 5 (2013), 18–26.

[2] Akoumianakis, D., Vidakis, N., Vellis, G., Kotsalis, D., Milolidakis, G., Plemenos, A., Akrivos, A., and Stefanakis, D. Transformable boundary artifacts for knowledge-based work in cross-organization virtual communities spaces. *Intelligent Decision Technologies 5*, 1 (2011), 65–82.

[3] Alavi, M., and Leidner, D. E. Knowledge management and knowledge management systems: Conceptual foundations and research issues. *MIS quarterly* (2001), 107–136.

[4] Allmark, P., Boote, J., Chambers, E., Clarke, A., McDonnell, A., Thompson, A., and Tod, A. M. Ethical issues in the use of in-depth interviews: Literature review and discussion. *Research Ethics 5*, 2 (2009), 48–54.

[5] Amit, R., and Schoemaker, P. J. H. Strategic assets and organizational rent. *Strategic Management Journal 14*, 1 (1993), 33–46.

[6] Andrews, K. M., and Delahaye, B. L. Influences on knowledge processes in organizational learning: The psychosocial filter. *Journal of Management studies 37*, 6 (2000), 797–810.

[7] Anh, N.-D., Cruzes, D. S., and Conradi, R. Dispersion, coordination and performance in global software teams: A systematic review.

In *Proceedings of the ACM-IEEE international symposium on Empirical software engineering and measurement* (2012), ACM, pp. 129–138.

[8] Annosi, M., Magnusson, M., Martini, A., and Appio, F. Social Conduct, Learning and Innovation: An Abductive Study of the Dark Side of Agile Software Development. *Creativity and Innovation Management 25*, 4 (2016), 515–535.

[9] Atuahene-Gima, K. Market orientation and innovation. *Journal of Business Research 35*, 2 (1996), 93 – 103.

[10] Aurum, A., Daneshgar, F., and Ward, J. Investigating Knowledge Management practices in software development organisations - An Australian experience. *Information and Software Technology 50*, 6 (2008), 511–533.

[11] Barney, J. Firm resources and sustained competitive advantage. *Advances in Strategic Management 17* (2000), 203–227. cited By 46.

[12] Barrett, M., and Oborn, E. Boundary object use in cross-cultural software development teams. *Human Relations 63*, 8 (2010), 1199–1221.

[13] Bechky, B. A. *Crossing occupational boundaries: Communication and learning on a production floor.* Stanford University, 1999.

[14] BECK, K., BEEDLE, M., BENNEKUM, A. V., COCKBURN, A., CUNNINGHAM, W., FOWLER, M., GRENNING, J., HIGHSMITH, J., HUNT, A., JEFFRIES, R., KERN, J., MARICK, B., MARTIN, R. C., MELLOR, S., SCHWABER, K., SUTHERLAND, J., and THOMAS, D. Manifesto for Agile Software Development, 2001.

[15] Begel, A., and Nagappan, N. Usage and perceptions of agile software development in an industrial context: An exploratory study. In *First International Symposium on Empirical Software Engineering and Measurement (ESEM 2007)* (Sep. 2007), pp. 255–264.

[16] Beleska-Spasova, E., Glaister, K., and Stride, C. Resource determinants of strategy and performance: The case of british exporters. *Journal of World Business 47*, 4 (2012), 635–647. cited By 29.

[17] Bjarnason, E., and Sharp, H. The role of distances in requirements communication: a case study. *Requirements Engineering 22*, 1 (2017), 1–26.

[18] Bjørnson, F. O., and Dingsøyr, T. Knowledge management in software engineering: A systematic review of studied concepts, findings and research methods used. *Information and Software Technology 50*, 11 (2008), 1055–1068.

[19] Bjørnson, F. O., and Vestues, K. Knowledge sharing and process improvement in large-scale agile development. In *Proceedings of the Scientific Workshop Proceedings of XP2016* (2016), ACM, p. 7.

[20] Blomkvist, J. K., Persson, J., and Åberg, J. Communication through boundary objects in distributed agile teams. In *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems* (2015), pp. 1875–1884.

[21] Blomqvist, K. The many faces of trust. *Scandinavian journal of management 13*, 3 (1997), 271–286.

[22] Boden, A., and Avram, G. Bridging knowledge distribution-the role of knowledge brokers in distributed software development teams. In *2009 ICSE Workshop on Cooperative and Human Aspects on Software Engineering* (2009), IEEE, pp. 8–11.

[23] Bolisani, E., and Bratianu, C. Knowledge strategy planning: an integrated approach to manage uncertainty, turbulence, and dynamics. *Journal of Knowledge Management 21*, 2 (2017), 233–253.

[24] Bolisani, E., and Scarso, E. Strategic planning approaches to knowledge management: a taxonomy. *VINE 45*, 4 (nov 2015), 495–508.

[25] Borrego, G., Morán, A. L., Palacio, R. R., Vizcaíno, A., and García, F. O. Towards a reduction in architectural knowledge vaporization during agile global software development. *Information and software technology 112* (2019), 68–82.

[26] Bradley, J. H., and Hebert, F. J. The effect of personality type on team performance. *Journal of Management Development 16*, 5 (1997), 337–353.

[27] Briers, M., and Chua, W. F. The role of actor-networks and boundary objects in management accounting change: a field study of an implementation of activity-based costing. *Accounting, Organizations and Society 26*, 3 (2001), 237–269.

[28] Bryman, A. *Social Research Methods.* Oxford University Press, New York, NY, USA, 2001.

[29] Budwig, M., Jeong, S., and Kelkar, K. When user experience met agile: A case study. In *CHI '09 Extended Abstracts on Human Factors in Computing Systems* (New York, NY, USA, 2009), CHI EA '09, ACM, pp. 3075–3084.

[30] Cabral, A. Y., Ribeiro, M. B., Lemke, A. P., Silva, M. T., Cristal, M., and Franco, C. A case study of knowledge management usage in agile software projects. In *International Conference on Enterprise Information Systems* (2009), Springer, pp. 627–638.

[31] Camacho, J. J., Sanches-Torres, J., and Galvis-Lista, E. Understanding the process of knowledge transfer in software engineering: A systematic literature review. In *The International Journal of Soft Computing and Software Engineering [JSCSE]. Special Issue: The Proceeding of International Conference on Soft Computing and Software Engineering 2013 [SCSE'13], Doi: 10.7321/jscse. v3. n3. 33 e-ISSN: 2251* (2013), vol. 7545, pp. 219–229.

[32] Capodieci, A., Mainetti, L., and Manco, L. A case study to enable and monitor real it companies migrating from waterfall to agile. In *International Conference on Computational Science and Its Applications* (2014), Springer, pp. 119–134.

[33] Carlile, P. R. Transferring, translating, and transforming: An integrative framework for managing knowledge across boundaries. *Organization science 15*, 5 (2004), 555–568.

[34] Carstensen, P. H., and Sørensen, C. From the social to the systematic. *Computer Supported Cooperative Work (CSCW) 5*, 4 (Dec 1996), 387–413.

[35] Chaffee, E. E. Three models of strategy. *Academy of management review 10*, 1 (1985), 89–98.

[36] Chau, T., and Maurer, F. Tool support for inter-team learning in agile software organizations. In *Advances in Learning Software Organizations* (Berlin, Heidelberg, 2004), G. Melnik and H. Holz, Eds., Springer Berlin Heidelberg, pp. 98–109.

[37] Chow, V., and Leiringer, R. The translation of power: A study of boundary objects in public engagement processes. In *30th Annual ARCOM Conference Proceedings 2014* (2014), Association of Researchers in Construction Management (ARCOM).

[38] Chyi Lee, C., Yang, J., and Ming Yu, L. The knowledge value of customers and employees in product quality. *Journal of management development 20*, 8 (2001), 691–706.

[39] Cockburn, A., and Highsmith, J. Agile software development: The people factor. *Computer*, 11 (2001), 131–133.

[40] Cohen, J. A coefficient of agreement for nominal scales. *Educational and psychological measurement 20*, 1 (1960), 37–46.

[41] Cohendet, P., and Edward Steinmueller, W. The codification of knowledge: a conceptual and empirical exploration. *Industrial and corporate change 9*, 2 (2000), 195–209.

[42] Conboy, K. Agility from first principles: Reconstructing the concept of agility in information systems development. *Information systems research 20*, 3 (2009), 329–354.

[43] Cooper, H. M. Organizing knowledge syntheses: A taxonomy of literature reviews. *Knowledge in Society 1*, 1 (1988), 104.

[44] Corbin, J., and Strauss, A. *Basics of qualitative research: techniques and procedures for developing grounded theory.* Sage publications, California, United States, 2015.

[45] Corbin, J. M., and Strauss, A. Grounded theory research: Procedures, canons, and evaluative criteria. *Qualitative Sociology 13*, 1 (Mar 1990), 3–21.

[46] Cowan, R., and Foray, D. The economics of codification and the diffusion of knowledge. *Industrial and corporate change 6*, 3 (1997), 595–622.

[47] Currall, S. C., and Judge, T. A. Measuring trust between organizational boundary role persons. *Organizational behavior and Human Decision processes 64*, 2 (1995), 151–170.

[48] Curtis, B., Herb, K., and Neil, I. A field study of the software design process for large systems. *Commun. ACM 31*, 11 (1988), 1268–1287.

[49] Datta, P., and Acar, W. Software and human agents in knowledge codification. *Knowledge Management Research & Practice 8*, 1 (2010), 45–60.

[50] Davenport, T. H., Prusak, L., et al. *Working knowledge: How organizations manage what they know.* Harvard Business Press, 1998.

[51] de O. Melo, C., Cruzes, D. S., Kon, F., and Conradi, R. Interpretative case studies on agile team productivity and management. *Information and Software Technology 55*, 2 (2013), 412 – 427. Special Section: Component-Based Software Engineering (CBSE), 2011.

[52] Dikert, K., Paasivaara, M., and Lassenius, C. Challenges and success factors for large-scale agile transformations: A systematic literature review. *Journal of Systems and Software 119* (2016), 87 – 108.

[53] Dingsøyr, T., and Hanssen, G. K. Extending agile methods: postmortem reviews as extended feedback. In *International Workshop on Learning Software Organizations* (2002), Springer, pp. 4–12.

[54] Dingsøyr, T., Rolland, K., Moe, N. B., and Seim, E. A. Coordination in multi-team programmes: An investigation of the group mode in large-scale agile software development. *Procedia Computer Science 121* (2017), 123–128.

[55] Dingsoyr, T., and Royrvik, E. An empirical study of an informal knowledge repository in a medium-sized software consulting company. In *25th International Conference on Software Engineering, 2003. Proceedings.* (May 2003), pp. 84–92.

[56] Dirks, K. T., and Ferrin, D. L. The role of trust in organizational settings. *Organization Science 12*, 4 (July 2001), 450–467.

[57] Dorairaj, S., Noble, J., and Malik, P. Knowledge management in distributed agile software development. In *2012 Agile Conference* (Aug 2012), pp. 64–73.

[58] Doran, H. D. Agile knowledge management in practice. In *International Workshop on Learning Software Organizations* (2004), Springer, pp. 137–143.

[59] Dybå, T., and Dingsøyr, T. Empirical studies of agile software development: A systematic review. *Information and Software Technology 50*, 9-10 (2008), 833–859.

[60] Dyba, T., and Dingsoyr, T. What do we know about agile software development? *IEEE Software 26*, 5 (2009), 6–9.

[61] Ebert, C. Requirements before the requirements: understanding the upstream impacts. In *13th IEEE International Conference on Requirements Engineering (RE'05)* (2005), IEEE, pp. 117–124.

[62] Edward Steinmueller, W. Will new information and communication technologies improve the 'codification' of knowledge? *Industrial and Corporate Change 9*, 2 (06 2000), 361–376.

[63] Eldh, S., and Murphy, B. Code ownership perspectives. *IEEE software 32*, 6 (2015), 18–19.

[64] Ersoy, I. B., and Mahdy, A. M. Agile knowledge sharing. *International Journal of Software Engineering (IJSE) 6*, 1 (2015), 1–15.

[65] Ferraresi, A. A., Quandt, C. O., dos Santos, S. A., and Frega, J. R. Knowledge management and strategic orientation: leveraging innovativeness and performance. *Journal of knowledge management 16*, 5 (2012), 688–701.

[66] Flyvbjerg, B. Five misunderstandings about case-study research. *Qualitative inquiry 12*, 2 (2006), 219–245.

[67] Frigg, R., and Hartmann, S. Models in science. In *The Stanford Encyclopedia of Philosophy*, E. N. Zalta, Ed., summer 2018 ed. Metaphysics Research Lab, Stanford University, 2018.

[68] Ghobadi, S. What drives knowledge sharing in software development teams: A literature review and classification framework. *Information & Management 52*, 1 (2015), 82–97.

[69] Ghobadi, S., and Mathiassen, L. A model for assessing and mitigating knowledge sharing risks in agile software development. *Information Systems Journal* (2016), 1–33.

[70] Ghobadi, S., and Mathiassen, L. Perceived barriers to effective knowledge sharing in agile software teams. *Information Systems Journal 26*, 2 (2016), 95–125.

[71] Glazer, R. Measuring the knower: Towards a theory of knowledge equity. *California Management Review 40*, 3 (1998), 175 – 194.

[72] Gorschek, T., Gomes, A., Pettersson, A., and Torkar, R. Introduction of a process maturity model for market-driven product management and requirements engineering. *Journal of Software: Evolution and Process 24*, 1 (2012), 83–113.

[73] Grant, R. M. Toward a knowledge-based theory of the firm. *Strategic Management Journal 17*, S2 (1996), 109–122.

[74] Gustavsson, T. Visualizing inter-team coordination. In *Proceedings of the Evaluation and Assessment in Software Engineering*. 2020, pp. 306–311.

[75] Halawi, L. A., McCarthy, R. V., and Aronson, J. E. Knowledge management and the competitive strategy of the firm. *The learning organization 13*, 4 (2006), 384–397.

[76] Hall, M. Knowledge management and the limits of knowledge codification. *Journal of Knowledge Management 10*, 3 (2006), 117–126.

[77] Hall, T., Beecham, S., and Rainer, A. Requirements problems in twelve software companies: an empirical analysis. *IEE Proceedings - Software 149*, 5 (Oct 2002), 153–160.

[78] Hansen, M. T., Nohria, N., and Tierney, T. What's your strategy for managing knowledge. *The knowledge management yearbook 2000–2001 77*, 2 (1999), 106–116.

[79] Hao, J., Zhao, Q., Yan, Y., and Wang, G. A review of tacit knowledge: Current situation and the direction to go. *International Journal of Software Engineering and Knowledge Engineering 27*, 05 (2017), 727–748.

[80] Hayek, F. A. The use of knowledge in society. *The American economic review 35*, 4 (1945), 519–530.

[81] Henderson, K. Flexible sketches and inflexible data bases: Visual communication, conscription devices, and boundary objects in design engineering. *Science, Technology, & Human Values 16*, 4 (1991), 448–473.

[82] Hendriks, P. Why share knowledge? the influence of ict on the motivation for knowledge sharing. *Knowledge and Process Management 6*, 2 (1999), 91–100.

[83] Highsmith, J., and Cockburn, A. Agile software development: The business of innovation. *Computer 34*, 9 (2001), 120–127.

[84] Hislop, D. *Knowledge Management in Organizations: A Critical Introduction*. OUP Oxford, 2013.

[85] Hoda, R., Noble, J., and Marshall, S. Developing a grounded theory to explain the practices of self-organizing agile teams. *Empirical Software Engineering 17*, 6 (2012), 609–639.

[86] Huber, T. L., Winkler, M. A., Dibbern, J., and Brown, C. V. The use of prototypes to bridge knowledge boundaries in agile software development. *Information systems journal 30*, 2 (2020), 270–294.

[87] Hutchins, E. The social organization of distributed cognition. In *Perspectives on Socially Shared Cognition*, L. Resnick, L. B., M. John, S. Teasley, and D., Eds. American Psychological Association, 1991, pp. 283–307.

[88] Huvila, I. The politics of boundary objects: Hegemonic interventions and the making of a document. *Journal of the American Society for Information Science and Technology 62*, 12 (2011), 2528–2539.

[89] Ivarsson, M., and Gorschek, T. A method for evaluating rigor and industrial relevance of technology evaluations. *Empirical Software Engineering 16*, 3 (2011), 365–395.

[90] Izquierdo-Cortazar, D., Robles, G., Ortega, F., and Gonzalez-Barahona, J. M. Using software archaeology to measure knowledge loss in software projects due to developer turnover. In *2009 42nd Hawaii International Conference on System Sciences* (Jan 2009), pp. 1–10.

[91] Jain, R., Cao, L., Mohan, K., and Ramesh, B. Situated boundary spanning: An empirical investigation of requirements engineering practices in product family development. *ACM Trans. Manage. Inf. Syst. 5*, 3 (dec 2014).

[92] Jung, J. J. Semantic wiki-based knowledge management system by interleaving ontology mapping tool. *International Journal of Software Engineering and Knowledge Engineering 23*, 01 (2013), 51–63.

[93] Käpyaho, M., and Kauppinen, M. Agile requirements engineering with prototyping: A case study. In *Requirements Engineering Conference (RE), 2015 IEEE 23rd International* (2015), IEEE, pp. 334–343.

[94] Karlsen, J. T., Hagman, L., and Pedersen, T. Intra-project transfer of knowledge in information systems development firms. *Journal of Systems and Information Technology 13*, 1 (2011), 66–80.

[95] Karlsson, L., Åsa G. Dahlstedt, Regnell, B., och Dag, J. N., and Persson, A. Requirements engineering challenges in market-driven software development: An interview study with practitioners. *Information and Software Technology 49*, 6 (2007), 588 – 604. Qualitative Software Engineering Research.

[96] Kasunic, M. Designing an effective survey. Tech. rep., Carnegie-Mellon Univ Pittsburgh PA Software Engineering Inst, 2005.

[97] Kaya, N., and Patton, J. The effects of knowledge-based resources, market orientation and learning orientation on innovation performance: An empirical study of Turkish firms. *Journal of International Development 23*, 2 (2011), 204–219.

[98] Kern, E.-M., and Kersten, W. Framework for internet-supported inter-organizational product development collaboration. *Journal of Enterprise Information Management* (2007).

[99] Kimble, C., Grenier, C., and Goglio-Primard, K. Innovation and knowledge sharing across professional boundaries: Political interplay between boundary objects and brokers. *International Journal of Information Management 30*, 5 (2010), 437–444.

[100] Kitchenham, B. A., Budgen, D., and Brereton, P. *Evidence-based software engineering and systematic reviews*, vol. 4. CRC press, 2015.

[101] Kittlaus, H.-B., and Fricker, S. *Software Product Management: The ISPMA-Compliant Study Guide and Handbook.* Springer, 2017.

[102] KOELMANN, H. Perspectives on information technology artefacts in trust-related interactions. In *International Conference on Human-Computer Interaction* (2020), Springer, pp. 445–457.

[103] KOGUT, B., AND ZANDER, U. Knowledge of the firm, combinative capabilities, and the replication of technology. *Organization science 3*, 3 (1992), 383–397.

[104] KRUCHTEN, P. Contextualizing agile software development. *Journal of Software: Evolution and Process 25*, 4 (2013), 351–361.

[105] KUUSINEN, K., GREGORY, P., SHARP, H., BARROCA, L., TAYLOR, K., AND WOOD, L. Knowledge sharing in a large agile organisation: A survey study. In *International Conference on Agile Software Development* (2017), Springer, pp. 135–150.

[106] LAI, L. F. A knowledge engineering approach to knowledge management. *Information Sciences 177*, 19 (2007), 4072 – 4094.

[107] LANDIS, J. R., AND KOCH, G. G. The measurement of observer agreement for categorical data. *biometrics* (1977), 159–174.

[108] LANSING, J., AND SUNYAEV, A. Trust in cloud computing: Conceptual typology and trust-building antecedents. *ACM sigmis database: The database for advances in Information Systems 47*, 2 (2016), 58–96.

[109] LARIVI$\surd$®RE, V., PONTILLE, D., AND SUGIMOTO, C. R. Investigating the division of scientific labor using the Contributor Roles Taxonomy (CRediT). *Quantitative Science Studies 2*, 1 (04 2021), 111–128.

[110] LAVRAKAS, P. Encyclopedia of Survey Research Methods, 2008.

[111] LEE, C. P. Boundary negotiating artifacts: Unbinding the routine of boundary objects and embracing chaos in collaborative work. *Computer Supported Cooperative Work (CSCW) 16*, 3 (2007), 307–339.

[112] LEE, E. C. Forming to performing: Transitioning large-scale project into agile. In *Agile 2008 Conference* (2008), IEEE, pp. 106–111.

[113] LEHTOLA, L., KAUPPINEN, M., AND VÄHÄNIITTY, J. Strengthening the link between business decisions and re: Long-term product planning in software product companies. In *15th IEEE International Requirements Engineering Conference (RE 2007)* (2007), IEEE, pp. 153–162.

[114] Leigh Star, S. This is not a boundary object: Reflections on the origin of a concept. *Science, Technology, & Human Values 35*, 5 (2010), 601–617.

[115] Lenberg, P., Feldt, R., and Wallgren, L.-G. Towards a behavioral software engineering. In *Proceedings of the 7th International Workshop on Cooperative and Human Aspects of Software Engineering* (New York, NY, USA, 2014), CHASE 2014, ACM, pp. 48–55.

[116] Lenberg, P., Feldt, R., and Wallgren, L.-G. Behavioral software engineering: A definition and systematic literature review. *Journal of Systems and Software 107* (2015), 15 – 37.

[117] Leung, L. Validity, reliability, and generalizability in qualitative research. *Journal of family medicine and primary care 4*, 3 (2015), 324.

[118] Levina, Natalia; Vaast, E. Turning a community into a market: A practice perspective on information technology use in boundary spanning. *Journal of Management Information Systems 22*, 4 (2014), 13–37.

[119] Levina, N., and Vaast, E. The emergence of boundary spanning competence in practice: Implications for implementation and use of information systems. *MIS quarterly* (2005), 335–363.

[120] Levy, M., and Hazzan, O. Knowledge management in practice: The case of agile software development. In *2009 ICSE Workshop on Cooperative and Human Aspects on Software Engineering* (2009), IEEE, pp. 60–65.

[121] Li, J., Moe, N. B., and Dybå, T. Transition from a plan-driven process to scrum: A longitudinal case study on software quality. In *Proceedings of the 2010 ACM-IEEE International Symposium on Empirical Software Engineering and Measurement* (New York, NY, USA, 2010), ESEM '10, ACM, pp. 13:1–13:10.

[122] Lindvall, M., Muthig, D., Dagnino, A., Wallin, C., Stupperich, M., Kiefer, D., May, J., and Kahkonen, T. Agile software development in large organizations. *Computer 37*, 12 (2004), 26–34.

[123] Luttikhuis, E. O., de Lange, J., Lutters, E., and ten Klooster, R. Evolving product information in aligning product development decisions across disciplines. *Procedia CIRP 29* (2015), 573–578.

[124] MacCormack, A., and Verganti, R. Managing the sources of uncertainty: Matching process and context in software development. *Journal of Product Innovation Management 20*, 3 (2003), 217–232.

[125] Mahesh, K., and Suresh, J. Knowledge criteria for organization design. *Journal of Knowledge Management 13*, 4 (2009), 41–51.

[126] Marakas, G. M., Johnson, R. D., and Palmer, J. W. A theoretical model of differential social attributions toward computing technology: when the metaphor becomes the model. *International Journal of Human-Computer Studies 52*, 4 (2000), 719–750.

[127] Maruping, L. M., Zhang, X., and Venkatesh, V. Role of collective ownership and coding standards in coordinating expertise in software project teams. *European Journal of Information Systems 18*, 4 (2009), 355–371.

[128] Mayer, R. C., Davis, J. H., and Schoorman, F. D. An integrative model of organizational trust. *Academy of management review 20*, 3 (1995), 709–734.

[129] McChesney, I. R., and Gallagher, S. Communication and coordination practices in software engineering projects. *Information and Software Technology 46*, 7 (2004), 473 – 489.

[130] McKnight, D. H., Cummings, L. L., and Chervany, N. L. Initial trust formation in new organizational relationships. *Academy of Management review 23*, 3 (1998), 473–490.

[131] Melnik, G., and Maurer, F. Direct verbal communication as a catalyst of agile knowledge sharing. *Proceedings of the Agile Development Conference, ADC 2004* (2004), 21–31.

[132] Méndez Fernández, D., Böhm, W., Vogelsang, A., Mund, J., Broy, M., Kuhrmann, M., and Weyer, T. Artefacts in software engineering: a fundamental positioning. *Software & Systems Modeling 18*, 5 (2019), 2777–2786.

[133] Miles, M. B., Huberman, A. M., and Saldaña, J. *Qualitative data analysis: A methods sourcebook.* Sage publications, 2018.

[134] Miller, D., and Shamsie, J. The Resource-Based View of the Firm in Two Environments: The Hollywood Firm Studios from 1936-1965. *Academy of Management Journal 39*, 3 (1996), 519–543.

[135] Moody, D. L. The method evaluation model: a theoretical model for validating information systems design methods. In *European Conference on Information Systems 2003* (2003), Department of Information Systems, London School of Economics, pp. 1–17.

[136] Munir, H., Runeson, P., and Wnuk, K. A theory of openness for software engineering tools in software organizations. *Information and Software Technology 97* (2018), 26–45.

[137] Nerur, S., Mahapatra, R., and Mangalaraj, G. Challenges of migrating to agile methodologies. *Communications of the ACM 48*, 5 (2005), 72–78.

[138] Nieves, J., Quintana, A., and Osorio, J. Knowledge-based resources and innovation in the hotel industry. *International Journal of Hospitality Management 38* (2014), 65–73. cited By 36.

[139] Nonaka, I. A dynamic theory of organizational knowledge creation. *Organization science 5*, 1 (1994), 14–37.

[140] Nonaka, I., Takeuchi, H., and Umemoto, K. A theory of organizational knowledge creation. *International Journal of Technology Management 11*, 7-8 (1996), 833–845.

[141] Nonaka, I., Toyama, R., and Konno, N. Seci, ba and leadership: a unified model of dynamic knowledge creation. *Long range planning 33*, 1 (2000), 5–34.

[142] Nordberg, M. E. Managing code ownership. *IEEE software 20*, 2 (2003), 26–33.

[143] OECD. Entrepreneurship - enterprises by business size - oecd data.

[144] Olmos-Sánchez, K., and Rodas-Osollo, J. Requirements engineering based on knowledge management: Theoretical aspects and a practical proposal. *International Journal of Software Engineering and Knowledge Engineering 27*, 08 (2017), 1199–1233.

[145] Ouriques, R., Fagerholm, F., Mendez, D., and Bern, B. G. An investigation of causes and effects of trust in boundary artefacts. *Information and Software Technology 158* (2023), 107170.

[146] Ouriques, R., Wnuk, K., Berntsson Svensson, R., and Gorschek, T. Thinking strategically about knowledge management in agile software development. In *Product-Focused Software Process Improvement* (Cham, 2018), M. Kuhrmann, K. Schneider, D. Pfahl, S. Amasaki, M. Ciolkowski, R. Hebig, P. Tell, J. Klünder, and S. Küpper, Eds., Springer International Publishing, pp. 389–395.

[147] Ouriques, R., Wnuk, K., Gorschek, T., and Svensson, R. B. The role of knowledge-based resources in agile software development contexts. *Journal of Systems and Software 197* (2023), 111572.

[148] Ouriques, R., Wnuk, K., Svensson, R. B., and Gorschek, T. Knowledge management strategies and processes in agile software development: A systematic literature review. *International Journal of Software Engineering and Knowledge Engineering 29*, 3 (2019), 345–380.

[149] O'Raghallaigh, P., Lane, S., Adam, F., and Sammon, D. Difficult knowledge boundaries during requirements determination. *Journal of Decision Systems 29*, sup1 (2020), 270–284.

[150] Paasivaara, M., Durasiewicz, S., and Lassenius, C. Using scrum in distributed agile development: A multiple case study. In *2009 Fourth IEEE International Conference on Global Software Engineering* (2009), IEEE, pp. 195–204.

[151] Paasivaara, M., and Lassenius, C. Could global software development benefit from agile methods? In *2006 IEEE International Conference on Global Software Engineering (ICGSE'06)* (Oct 2006), pp. 109–113.

[152] Paravastu, N., Gefen, D., and Creason, S. B. Understanding trust in it artifacts: An evaluation of the impact of trustworthiness and trust on satisfaction with antiviral software. *ACM SIGMIS Database: the DATABASE for Advances in Information Systems 45*, 4 (2014), 30–50.

[153] Paredes, J., Anslow, C., and Maurer, F. Information visualization for agile software development. In *2014 Second IEEE Working Conference on Software Visualization* (Sep. 2014), pp. 157–166.

[154] Penrose, E., and Penrose, E. T. *The Theory of the Growth of the Firm.* Oxford university press, 2009.

[155] Popay, J., Roberts, H., Sowden, A., Petticrew, M., Arai, L., Rodgers, M., Britten, N., Roen, K., and Duffy, S. Guidance on the conduct of narrative synthesis in systematic reviews. *A product from the ESRC methods programme Version 1* (2006), b92.

[156] Porter, M. E. *Competitive advantage: Creating and sustaining superior performance.* Simon and Schuster, 2008.

[157] Regnell, B., and Brinkkemper, S. Market-driven requirements engineering for software products. In *Engineering and managing software requirements.* Springer, 2005, pp. 287–308.

[158] Robillard, P. N. The role of knowledge in software development. *Communications of the ACM 42*, 1 (1999), 87–92.

[159] Robson, C., and McCartan, K. *Real World Research.* John Wiley & Sons, Ltd, London, UK, 2011.

[160] Runeson, P., and Höst, M. Guidelines for conducting and reporting case study research in software engineering. *Empirical software engineering 14*, 2 (2009), 131–164.

[161] Rus, I., and Lindvall, M. Knowledge management in software engineering. *IEEE Software 19*, 3 (2002), 26–38.

[162] Ryan, S., and O'Connor, R. V. Acquiring and sharing tacit knowledge in software development teams: An empirical study. *Information and Software Technology 55*, 9 (2013), 1614 – 1624.

[163] Saito, S., Iimura, Y., Massey, A. K., and Antón, A. I. Discovering undocumented knowledge through visualization of agile software development activities. *Requirements Engineering 23*, 3 (Sep 2018), 381–399.

[164] Santos, V., Goldman, A., and de Souza, C. Fostering effective inter-team knowledge sharing in agile software development. *Empirical Software Engineering 20*, 4 (2015), 1006–1051.

[165] Sapsed, J., and Salter, A. Postcards from the edge: local communities, global programs and boundary objects. *Organization studies 25*, 9 (2004), 1515–1534.

[166] Scheepers, R., Venkitachalam, K., and Gibbs, M. R. Knowledge strategy in organizations: refining the model of hansen, nohria and tierney. *The Journal of Strategic Information Systems 13*, 3 (2004), 201–222.

[167] Seaman, C. B. Qualitative methods in empirical studies of software engineering. *IEEE Transactions on software engineering 25*, 4 (1999), 557–572.

[168] Sedano, T., Ralph, P., and Péraire, C. Software development waste. In *Proceedings of the 39th International Conference on Software Engineering* (Piscataway, NJ, USA, 2017), ICSE '17, IEEE Press, pp. 130–140.

[169] Sirmon, D., and Hitt, M. Contingencies within dynamic managerial capabilities: Interdependent effects of resource investment and deployment on firm performance. *Strategic Management Journal 30*, 13 (2009), 1375–1394. cited By 159.

[170] Smite, D., Christensen, E. L., Tell, P., and Russo, D. The future workplace: Characterizing the spectrum of hybrid work arrangements for software teams. *IEEE Software 40*, 2 (2023), 34–41.

[171] Soliman, F., and Spooner, K. Strategies for implementing knowledge management: role of human resources management. *Journal of Knowledge Management 4*, 4 (2000), 337–345.

[172] Soomro, A. B., Salleh, N., Mendes, E., Grundy, J., Burch, G., and Nordin, A. The effect of software engineers' personality traits on team climate and performance: A systematic literature review. *Information and Software Technology 73* (2016), 52 – 65.

[173] Sørensen, C., and Lundh-Snis, U. Innovation through knowledge codification. *Journal of Information Technology 16*, 2 (2001), 83–97.

[174] Star, S. L. Chapter 2 - the structure of ill-structured solutions: Boundary objects and heterogeneous distributed problem solving. In *Distributed Artificial Intelligence*, L. Gasser and M. N. Huhns, Eds. Morgan Kaufmann, San Francisco (CA), 1989, pp. 37–54.

[175] Star, S. L., and Griesemer, J. R. Institutional ecology,translations' and boundary objects: Amateurs and professionals in berkeley's museum of vertebrate zoology, 1907-39. *Social studies of science 19*, 3 (1989), 387–420.

[176] Steen, O. Practical knowledge and its importance for software product quality. *Information and Software Technology 49*, 6 (2007), 625–636.

[177] Stettina, C. J., and Heijstek, W. Necessary and neglected? an empirical study of internal documentation in agile software development teams. In *Proceedings of the 29th ACM International Conference on Design of Communication* (New York, NY, USA, 2011), SIGDOC '11, Association for Computing Machinery, p. 159–166.

[178] Stettina, C. J., Heijstek, W., and Fægri, T. E. Documentation work in agile teams: The role of documentation formalism in achieving a sustainable practice. In *2012 Agile Conference* (Aug 2012), pp. 31–40.

[179] Stol, K., Ralph, P., and Fitzgerald, B. Grounded theory in software engineering research: A critical review and guidelines. In *2016 IEEE/ACM 38th International Conference on Software Engineering (ICSE)* (May 2016), pp. 120–131.

[180] Strode, D. E., Huff, S. L., Hope, B., and Link, S. Coordination in co-located agile software development projects. *Journal of Systems and Software 85*, 6 (2012), 1222–1238.

[181] Subrahmanian, E., Monarch, I., Konda, S., Granger, H., Milliken, R., Westerberg, A., et al. Boundary objects and prototypes at the interfaces of engineering design. *Computer Supported Cooperative Work (CSCW) 12*, 2 (2003), 185–203.

[182] Sveiby, K.-E. A knowledge-based theory of the firm to guide in strategy formulation. *Journal of intellectual capital 2*, 4 (2001), 344–358.

[183] Swedish Research Council. *Good Research Practice.* Swedish Research Council, Stockholm, Sweden, 2017.

[184] Szulanski, G., Cappetta, R., and Jensen, R. J. When and how trustworthiness matters: Knowledge transfer and the moderating effect of causal ambiguity. *Organization Science 15* (2004), 600–613.

[185] Teece, D. J. Strategies for managing knowledge assets: the role of firm structure and industrial context. *Long range planning 33*, 1 (2000), 35–54.

[186] Thatcher, J. B., McKnight, D. H., Baker, E. W., Arsal, R. E., and Roberts, N. H. The role of trust in postadoption it exploration: An

empirical examination of knowledge management systems. *IEEE Transactions on Engineering Management 58*, 1 (2010), 56–70.

[187] TOM, E., AURUM, A., AND VIDGEN, R. An exploration of technical debt. *Journal of Systems and Software 86*, 6 (2013), 1498–1516.

[188] VÄHÄNIITTY, J., AND RAUTIAINEN, K. T. Towards a conceptual framework and tool support for linking long-term product and business planning with agile software development. In *Proceedings of the 1st international workshop on Software development governance* (2008), ACM, pp. 25–28.

[189] VAN LOGGEM, B., AND VAN DER VEER, G. C. A documentation-centred approach to software design, development and deployment. In *Building Bridges: HCI, Visualization, and Non-formal Modeling* (Berlin, Heidelberg, 2014), A. Ebert, G. C. van der Veer, G. Domik, N. D. Gershon, and I. Scheler, Eds., Springer Berlin Heidelberg, pp. 188–200.

[190] VIDOTTO, G., MASSIDDA, D., NOVENTA, S., AND VICENTINI, M. Trusting beliefs: A functional measurement study. *Psicologica: International Journal of Methodology and Experimental Psychology 33*, 3 (2012), 575–590.

[191] VOM BROCKE, J., SIMONS, A., RIEMER, K., NIEHAVES, B., PLATTFAUT, R., AND CLEVEN, A. Standing on the shoulders of giants: Challenges and recommendations of literature search in information systems research. *Communications of the association for information systems 37*, 1 (2015), 9.

[192] WALCZAK, S. Organizational knowledge management structure. *The Learning Organization 12*, 4 (2005), 330–339.

[193] WALLIN, C., EKDAHL, F., AND LARSSON, S. Integrating business and software development models. *IEEE software 19*, 6 (2002), 28–33.

[194] WEST, D., GRANT, T., GERUSH, M., AND D'SILVA, D. Agile development: Mainstream adoption has changed agility. *Forrester Research 2*, 1 (2010), 41.

[195] WHETTEN, D. A. What constitutes a theoretical contribution? *Academy of management review 14*, 4 (1989), 490–495.

[196] Whitson, J. R. Voodoo software and boundary objects in game development: How developers collaborate and conflict with game engines and art tools. *new media & society 20*, 7 (2018), 2315–2332.

[197] Williams, L., and Cockburn, A. Agile software development: it's about feedback and change. *Computer 36*, 6 (June 2003), 39–43.

[198] Wilson, T. D. The nonsense of knowledge management. *Information research 8*, 1 (2002), 8–1.

[199] Wohlin, C. Guidelines for snowballing in systematic literature studies and a replication in software engineering. In *Proceedings of the 18th international conference on evaluation and assessment in software engineering* (2014), Citeseer, p. 38.

[200] Wohlin, C., Runeson, P., Höst, M., Ohlsson, M. C., Regnell, B., and Wesslén, A. *Experimentation in software engineering.* Springer Science & Business Media, 2012.

[201] Wohlrab, R., Pelliccione, P., Knauss, E., and Larsson, M. Boundary objects in agile practices: Continuous management of systems engineering artifacts in the automotive domain. In *Proceedings of the 2018 International Conference on Software and System Process* (New York, NY, USA, 2018), ICSSP '18, Association for Computing Machinery, p. 31–40.

[202] Wohlrab, R., Pelliccione, P., Knauss, E., and Larsson, M. Boundary objects and their use in agile systems engineering. *Journal of Software: Evolution and Process 31*, 5 (2019), e2166.

[203] Yanzer Cabral, A. R., Ribeiro, M. B., and Noll, R. P. Knowledge management in agile software projects: A systematic review. *Journal of Information & Knowledge Management 13*, 01 (2014), 1450010.

[204] Zabardast, E., Gonzalez-Huerta, J., and Tanveer, B. Ownership vs contribution: Investigating the alignment between ownership and contribution. In *2022 IEEE 19th International Conference on Software Architecture Companion (ICSA-C)* (2022), pp. 30–34.

[205] Zahedi, M., Shahin, M., and Babar, M. A. A systematic review of knowledge sharing challenges and practices in global software development. *International Journal of Information Management 36*, 6 (2016), 995–1019.

[206] Zaitsev, A., Gal, U., and Tan, B. Boundary objects and change in agile projects. ACIS.

[207] Zurba, M., Maclean, K., Woodward, E., and Islam, D. Amplifying indigenous community participation in place-based research through boundary work. *Progress in Human Geography 43*, 6 (2019), 1020–1043.

Printed by Media-Tryck, Lund 2023 NORDIC SWAN ECOLABEL 3041 0903

Context: Many software companies adopt Agile Software Development (ASD) principles through various methods, aiming to respond rapidly to market changes or internal transformations. As agile methods prioritise intense communication between people over documentation to bring more flexibility and readiness to welcome changes, the pressure on how knowledge is shared and applied increases. Many knowledge resources remain intangible in these contexts, requiring lots of effort to understand what should remain tacit and what should be captured explicitly as artefacts.

Objective: This thesis aims to contribute to a better understanding of knowledge resources in agile software project environments and provide guidance on effectively managing them.

Method: We follow mostly a qualitative approach. We adhere to social constructivism research, which notes that social phenomena undergo constant changes and are affected by human interaction. As qualitative and quantitative methods of investigation, we utilised literature reviews, grounded theory, survey and a case study.

Results: We synthesised evidence from the literature to show the proportions of knowledge management practices utilised in ASD environments and the knowledge process they focus on. Through a grounded theory study, we identified Knowledge-based Resources (KBRs) that support changes in agile environments in the Knowledge-push theory. In this study, we identified inefficiencies in converting KBRs into Property-based Resources (PBRs). This evidence led us to a case study in which we investigated the causes and effects of trust in Boundary Artefacts (BAs), a specialisation of PBRs. The results have contributed to understanding the favourable factors that make stakeholders feel confident in utilising BAs and pointed to the implication of decreased trust in software projects. Such negative implications can be mitigated by applying our developed and validated guideline that supports the creation of BAs in software engineering, which was perceived as being able to increase the trustworthiness of BAs. Lastly, we gathered the evidence that we collected through this doctoral journey and offered a simplified discussion about knowledge resources in an agile context.

Conclusions: We clarify the concept of KBRs, identify them, and explain how they support changes in agile contexts. In this process, we uncover the inefficiencies in converting KBRs into PBRs and support their management in software projects. For example, (i) understanding how trust aspects such as reliability, predictability, and functionality affect practitioners' confidence in BAs, (ii) providing a structured guideline that helps practitioners create BAs, (iii) incorporating more formal practices to manage BAs that do not necessarily abandon agile flexibility to deal with changes.

Blekinge Institute of Technology
Doctoral Dissertation Series No. 2023:14
ISSN: 1653-2090
ISBN: 978-91-7295-467-0