The role of knowledge-based resources in Agile Software Development contexts

The Journal of Systems & Software 197 (2023) 111572

Contents lists available at ScienceDirect

The Journal of Systems & Software

journal homepage: www.elsevier.com/locate/jss

In practice

The role of knowledge-based resources in Agile Software Development contexts☆

Raquel Ouriques [a,*], Krzysztof Wnuk [a], Tony Gorschek [a], Richard Berntsson Svensson [b]

[a] Department of Software Engineering, Blekinge Institute of Technology, Karskrona, 37179, Sweden
[b] Department of Computer Science and Engineering at Chalmers | University of Gothenburg, Gothenburg, 41296, Sweden

ARTICLE INFO

Article history:
Received 12 July 2021
Received in revised form 20 June 2022
Accepted 24 November 2022
Available online 28 November 2022

Keywords:
Knowledge-based resources
Agile software development
Grounded theory
Software development
Knowledge management

ABSTRACT

The software value chain is knowledge-based since it is highly dependant on people. Consequently, a lack of practice in managing knowledge as a resource may jeopardise its application in software development. Knowledge-Based Resources (KBRs) relate to employees' intangible knowledge that is deemed to be valuable to a company's competitive advantage. In this study, we apply a grounded theory approach to examine the role of KBRs in Agile Software Development (ASD). To this aim, we collected data from 18 practitioners from five companies. We develop the Knowledge-Push theory, which explains how KBRs boost the need for change in ASD. Our results show that the practitioners who participated in the study utilise, as primary strategies, task planning, resource management, and social collaboration. These strategies are implemented through the team environment and settings and incorporate an ability to codify and transmit knowledge. However, this process of codification is non-systematic, which consequently introduces inefficiency in the domain of knowledge resource utilisation, resulting in potential knowledge waste. This inefficiency can generate negative implications for 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.

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 (Sedano et al., 2017).

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 (Strode et al., 2012). ASD consequently relies on self-organised teams that utilise informal communication and tacit knowledge (Bjørnson and Dingsøyr, 2008) that is shared in an *ad hoc* manner among the team members (Cockburn and Highsmith, 2001; Rus and Lindvall, 2002). This process may take place in both co-located and distributed environments (Ghobadi and Mathiassen, 2016; Melnik and Maurer, 2004). Although tacit knowledge is relevant for companies and crucial for innovation (Nonaka, 1994), it is only revealed when it is applied (Grant, 1996). Such knowledge is acquired via experience and transferred among people. This can prove to be costly, slow, and uncertain (Kogut and Zander, 1992).

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 (Barney, 2000; Glazer, 1998). 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 (Steen, 2007).

A company's competitive strategy is substantially dependent on how the company maximises its produced value by allocating its resources (Barney, 2000). 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 (Miller and Shamsie, 1996). 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.

☆ Editor: Marcos Kalinowski.

* Corresponding author.
E-mail addresses: rou@bth.se (R. Ouriques), krw@bth.se (K. Wnuk), tgo@bth.se (T. Gorschek), richard@cse.gu.se (R.B. Svensson).

https://doi.org/10.1016/j.jss.2022.111572

Because the software value chain is knowledge-based (Rus and Lindvall, 2002), 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 (Miller and Shamsie, 1996; Conboy, 2009).

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 (McChesney and Gallagher, 2004; Ouriques et al., 2019). 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 (Rus and Lindvall, 2002; Bjørnson and Dingsøyr, 2008), 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 2, we present a brief background to our study and related work. Section 3 describes the research method. In Section 4, we present the Knowledge-Push theory, along with a description of each category that the theory relates to. In Section 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 6, we discuss the threats to the validity of our study. Finally, in Section 7, we present our concluding remarks.

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.

2.1. Agile software development

Flexibility in responding to change (Williams and Cockburn, 2003) is the focal point of Agile Software Development, an umbrella term that includes several methods and frameworks that inform software development practices (Conboy, 2009; Dybå and Dingsøyr, 2008; Beck et al., 2001; West et al., 2010). 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 (Beck et al., 2001).

Dybå and Dingsøyr (2008) 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. (2016) 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. (2010) 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 (Rus and Lindvall, 2002). 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 (Cockburn and Highsmith, 2001).

Previous literature on the subject has indicated that companies who adopt ASD rely on tacit knowledge to a large extent (Bjørnson and Dingsøyr, 2008). Furthermore, they rely on informal means of communication as they share this knowledge between people, in both co-located and distributed ASD contexts (Ghobadi and Mathiassen, 2016; Melnik and Maurer, 2004; Dikert et al., 2016). Although tacit knowledge is recognised as a relevant resource for companies and crucial for innovation (Nonaka, 1994), it can only be revealed when it is applied (Grant, 1996). 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 (Kogut and Zander, 1992).

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 (Alavi and Leidner, 2001).

The nature of knowledge has been discussed extensively among philosophers (Grant, 1996). 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" (Alavi and Leidner, 2001; Nonaka, 1994).

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 (Nonaka et al., 2000).

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 (Hislop, 2013; Grant, 1996; Barney, 2000).

This knowledge management work can occur through four knowledge processes (Alavi and Leidner, 2001; Cowan and Foray, 1997):

- 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.

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 (Barney, 2000).

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 (Miller and Shamsie, 1996).

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 (Atuahene-Gima, 1996; Sirmon and Hitt, 2009). 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 (Miller and Shamsie, 1996).

Previous research has provided evidence that KBRs positively contribute to return on sale, operating profit, and market share (Miller and Shamsie, 1996). 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 (Nieves et al., 2014).

Kaya and Patton (2011) 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 (Barney, 2000; Beleska-Spasova et al., 2012). 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 (Amit and Schoemaker, 1993).

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 (Miller and Shamsie, 1996). However, as far as we know, knowledge-based resources in software engineering or in ASD have not been addressed in the literature to date.

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. 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 (Ouriques et al., 2019, 2018) and our discussions and brainstorming sessions with three companies. Two RQs are addressed below:

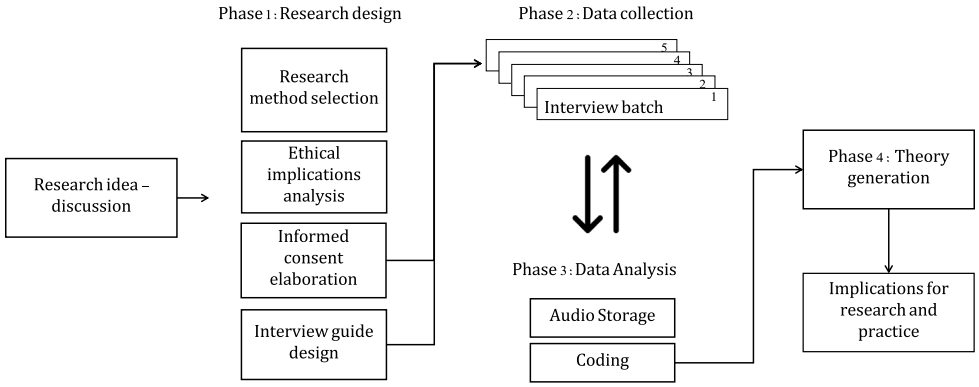

Fig. 1. Research process overview.

- 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 (Grant, 1996).

We conducted a qualitative study so as to efficiently capture data regarding the complex phenomena of human behaviour (Seaman, 1999; Hoda et al., 2012).

We followed the principles of Grounded Theory as a method for theory generation (Corbin and Strauss, 1990, 2015). 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 (Bryman, 2001). Additionally, we adopted a theoretical perspective that recognises that knowledge is something that is socially constructed (Nonaka, 1994).

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 (Seaman, 1999). The method of analysis that is detailed by Corbin and Strauss (2015, 1990) 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.

Fig. 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. (2016), which provide a framework for reporting on grounded theory in software engineering.

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 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.

3.1.1. Sampling of practitioners

Practitioner selection was performed by means of a convenience sampling (Lavrakas, 2008) that focused on participants whose activities were related to coordination. As we proceeded with our interviews, we applied a process of 'theoretical sampling' (Corbin and Strauss, 2015). 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 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 (OECD, 2022).

3.1.2. 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.

Table 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

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 schema. However, the documents served as contextualisation, 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 min. 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.

3.1.3. 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).

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

3.1.4. 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 (Swedish Research Council, 2017).

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 (Allmark et al., 2009).

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.

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.

3.2. Phase 3 — Data analysis phase

The data collection process and the analysis of said data were interrelated (Corbin and Strauss, 1990). 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.

Concepts	Category
Identify what knowledge to keep in a systematic way	
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	**Ability to systematise and transmit knowledge**
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	

Fig. 2. Example of the emergence of a concept.

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"	**Conception of a knowledge retention structure**
"we use different tools for support. We try to store knowledge in Jira, but sometimes we find the same thing in two different places"	
"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"	**Knowing what others know**
"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"	

Fig. 3. Example of the emergence of a category.

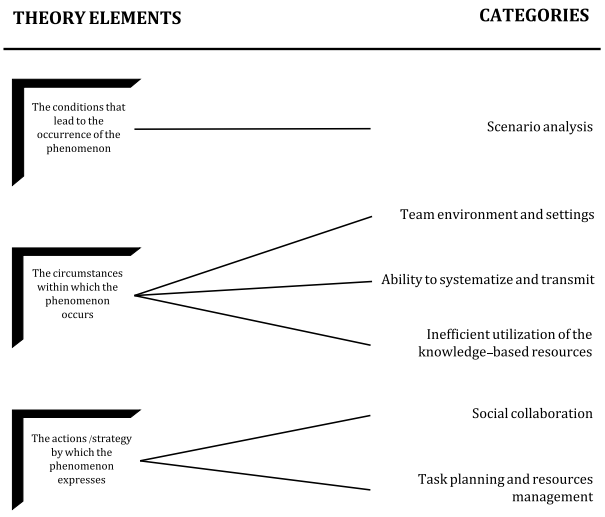

Fig. 4. Illustration of the diagramming of the categories.

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 (Corbin and Strauss, 2015), 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 (Corbin and Strauss, 1990). 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 Fig. 2), we observed reoccurring 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.

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 Fig. 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 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 (Corbin and Strauss, 1990). Fig. 4 displays this 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 (Corbin and Strauss, 1990). 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 (Frigg and Hartmann, 2018). 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

Table 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 - combination of interpersonal skills	- identify what knowledge to keep in a systematic way - experience transfer	- meaningless search in a database - 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 - representing knowledge efficiently in an artefact - conception of a knowledge retention structure - ability to integrate tool and coordination skills - how architectural knowledge is disseminated	- waste substantial time

generated categories with previous research. In this process, we related our research findings to the existing literature to validate our theory (see Section 3.3), to strengthen our understanding of each category, and to support our discussion (detailed in Section 5).

3.3. Theory evaluation

Corbin and Strauss (1990) 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 (Corbin and Strauss, 1990). We complemented this validation by following the guidelines for conducting Grounded Theory as proposed by Stol et al. (2016).

We evaluated the empirical basis of our findings by means of seven criteria (Corbin and Strauss, 1990):

- 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

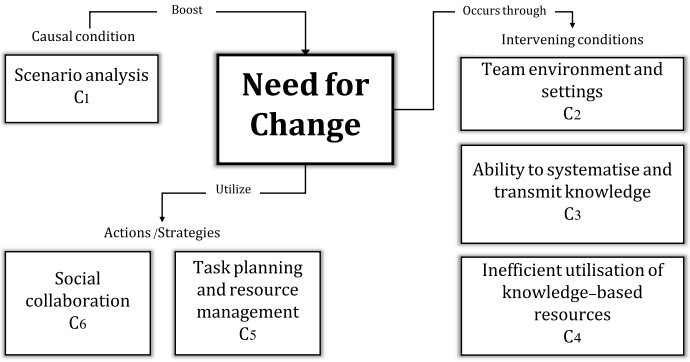

Fig. 5. The Knowledge-push Theory.

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 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 5.1).

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 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 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

4.1. The Knowledge-Push theory

Fig. 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 (1990):

- 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).
- 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 (Barney, 2000). 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.

Table 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.

4.2. 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.

4.2.1. 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".

4.3. 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.

4.3.1. 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 co-existence 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 (Nonaka, 1994). 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.

4.3.2. Ability to systematise 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 systematised 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".

4.3.3. Inefficient utilisation 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 (2002) 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 (Melo et al., 2013; Izquierdo-Cortazar et al., 2009; Ersoy and Mahdy, 2015; Sedano et al., 2017), but our findings also reveal additional inefficiencies, including the redesign of solutions due to an unawareness of knowledge sources. In Section 5, we expand on this discussion by providing a number of potential implications that stem from these inefficiencies.

4.4. Actions/strategy

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

4.4.1. 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".

4.4.2. 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.

5. Discussion

This section discusses in Section 5.1 how our findings relate to and add to the existing literature on KBRs and software engineering. In Section 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.

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 (Miller and Shamsie, 1996; Kaya and Patton, 2011; Nieves et al., 2014).

In periods of stability, companies benefit from property-based resources. However, in changing and unpredictable environments, KBRs have contributed to increased profits and sales (Miller and Shamsie, 1996). 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 (Kittlaus and Fricker, 2017; MacCormack and Verganti, 2003), requirements engineering (Karlsson et al., 2007; Hall et al., 2002; Curtis et al., 1988), and market-driven software development (Gorschek et al., 2012). 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 (Gorschek et al., 2012) 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 (Nieves et al., 2014). 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 (Ryan and O'Connor, 2013; Ouriques et al., 2019; Bjørnson and Dingsøyr, 2008; Rus and Lindvall, 2002; Bradley and Hebert, 1997). 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 (Bradley and Hebert, 1997). 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 (Lenberg et al., 2014, 2015; Soomro et al., 2016).

Encouraging collective thinking can be viewed as an alternative to triggering the cognitive processes for combining knowledge (Nonaka et al., 1996). 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 (Ouriques et al., 2019; Dorairaj et al., 2012; Chau and Maurer, 2004; Karlsen et al., 2011; Kuusinen et al., 2017). 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 (Carstensen and Sørensen, 1996; Sørensen and Lundh-Snis, 2001; Dingsøyr and Royrvik, 2003; Edward Steinmueller, 2000; Datta and Acar, 2010). 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 (Li et al., 2010; Käpyaho and Kauppinen, 2015; Budwig et al., 2009; Begel and Nagappan, 2007). Borrego et al. (2019) argue that poor design documentation results in the loss of architectural knowledge, which they define as knowledge vaporisation. 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 (Paredes et al., 2014). Because KBRs might affect a company's performance (Beleska-Spasova et al., 2012), 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.

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 (Ouriques et al., 2019).

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 4.4.1) 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 (Ouriques et al., 2018).

Knowledge management practices should be developed with the aim of achieving a purpose. Furthermore, they need organisational support and stimuli (Santos et al., 2015). 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 (Nonaka et al., 2000).

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 (Nonaka et al., 2000). Proper leadership can mediate processes at each company level by aiming to enhance knowledge creation, storage, sharing/transfer, and application (Ouriques et al., 2019).

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 Fig. 6.

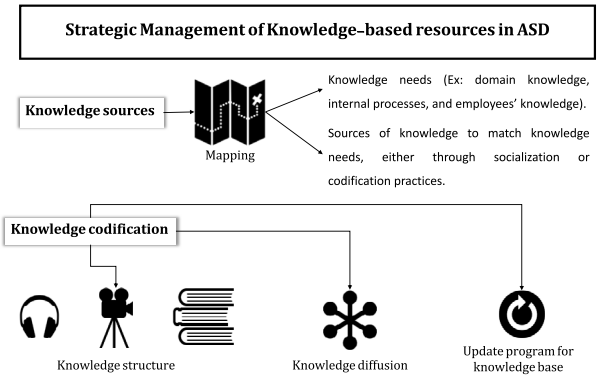

Fig. 6. Recommendations for managing knowledge.

5.2.1. 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. (2012) 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 behaviour can contribute to documentation debt (Dybå and Dingsøyr, 2008; Annosi et al., 2016; Tom et al., 2013).

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 (Saito et al., 2018).

Whilst knowledge codification can be costly when appropriately planned and implemented, it can positively affect innovation, economic growth, and knowledge creation (Aurum et al., 2008; Cohendet and Edward Steinmueller, 2000).

Cohendet and Edward Steinmueller (2000) 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 (Hendriks, 1999).

- *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 (2006), 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 (Lai, 2007). 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 (Alavi and Leidner, 2001) can result in reduced time wasted and reduced deployment delays by speeding up the knowledge retrieval process.

5.2.2. 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 (Rus and Lindvall, 2002).

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 5.2.1), 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 Andrews and Delahaye (2000) and Szulanski et al. (2004).

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' (Dirks and Ferrin, 2001). A high degree of receptiveness can lead to reduced knowledge exchange costs (Currall and Judge, 1995).

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 (2002), 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 (Soliman and Spooner, 2000). 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.

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. (2012), 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 (Flyvbjerg, 2006) 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 (Corbin and Strauss, 1990). 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).

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).

Although Grounded Theory provides ways for validating theory generation (see Section 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 (Wohlin et al., 2012) 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.

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 Section 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?

CRediT authorship contribution statement

Raquel Ouriques: Conceptualization, Methodology, Investigation, Resources, Data curation, Validation, Writing – original draft, Writing – review & editing, Visualization. **Krzysztof Wnuk:** Investigation, Writing – original draft, Writing – review & editing, Supervision. **Tony Gorschek:** Supervision, Reviewing, Funding acquisition. **Richard Berntsson Svensson:** Writing – reviewing.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgement

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

Appendix A. Supplementary data

Supplementary material related to this article can be found online at https://doi.org/10.1016/j.jss.2022.111572.

References

Alavi, M., Leidner, D.E., 2001. Knowledge management and knowledge management systems: Conceptual foundations and research issues. MIS Q. 25, 107–136.

Allmark, P., Boote, J., Chambers, E., Clarke, A., McDonnell, A., Thompson, A., Tod, A.M., 2009. Ethical issues in the use of in-depth interviews: Literature review and discussion. Res. Ethics 5 (2), 48–54. http://dx.doi.org/10.1177/174701610900500203.

Amit, R., Schoemaker, P.J.H., 1993. Strategic assets and organizational rent. Strateg. Manag. J. 14 (1), 33–46. http://dx.doi.org/10.1002/smj.4250140105.

Andrews, K.M., Delahaye, B.L., 2000. Influences on knowledge processes in organizational learning: the psychosocial filter. J. Manag. Stud. 37 (6), 797–810. http://dx.doi.org/10.1111/1467-6486.00204.

Annosi, M., Magnusson, M., Martini, A., Appio, F., 2016. Social conduct, learning and innovation: An abductive study of the dark side of agile software development. Creat. Innov. Manag. 25 (4), 515–535. http://dx.doi.org/10.1111/caim.12172.

Atuahene-Gima, K., 1996. Market orientation and innovation. J. Bus. Res. 35 (2), 93–103. http://dx.doi.org/10.1016/0148-2963(95)00051-8.

Aurum, A., Daneshgar, F., Ward, J., 2008. Investigating knowledge management practices in software development organisations - An Australian experience. Inf. Softw. Technol. 50 (6), 511–533. http://dx.doi.org/10.1016/j.infsof.2007.05.005.

Barney, J., 2000. Firm resources and sustained competitive advantage. Adv. Strateg. Manag. 17, 203–227. http://dx.doi.org/10.1016/S0742-3322(00)17018-4, cited By 46.

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., Thomas, D., 2001. Manifesto for agile software development.

Begel, A., Nagappan, N., 2007. 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). pp. 255–264. http://dx.doi.org/10.1109/ESEM.2007.12.

Beleska-Spasova, E., Glaister, K., Stride, C., 2012. Resource determinants of strategy and performance: The case of British exporters. J. World Bus. 47 (4), 635–647. http://dx.doi.org/10.1016/j.jwb.2011.09.001, cited By 29.

Bjørnson, F.O., Dingsøyr, T., 2008. Knowledge management in software engineering: A systematic review of studied concepts, findings and research methods used. Inf. Softw. Technol. 50 (11), 1055–1068. http://dx.doi.org/10.1016/j.infsof.2008.03.006.

Borrego, G., Morán, A.L., Palacio, R.R., Vizcaíno, A., García, F.O., 2019. Towards a reduction in architectural knowledge vaporization during agile global software development. Inf. Softw. Technol. 112, 68–82. http://dx.doi.org/10.1016/j.infsof.2019.04.008.

Bradley, J.H., Hebert, F.J., 1997. The effect of personality type on team performance. J. Manag. Dev. 16 (5), 337–353. http://dx.doi.org/10.1108/02621719710174525.

Bryman, A., 2001. Social Research Methods. Oxford University Press, New York, NY, USA.

Budwig, M., Jeong, S., Kelkar, K., 2009. When user experience met agile: A case study. In: CHI '09 Extended Abstracts on Human Factors in Computing Systems. In: CHI EA '09, ACM, New York, NY, USA, pp. 3075–3084. http://dx.doi.org/10.1145/1520340.1520434.

Carstensen, P.H., Sørensen, C., 1996. From the social to the systematic. Comput. Support. Coop. Work (CSCW) 5 (4), 387–413. http://dx.doi.org/10.1007/BF00136712.

Chau, T., Maurer, F., 2004. Tool support for inter-team learning in agile software organizations. In: Melnik, G., Holz, H. (Eds.), Advances in Learning Software Organizations. Springer Berlin Heidelberg, Berlin, Heidelberg, pp. 98–109.

Cockburn, A., Highsmith, J., 2001. Agile software development: The people factor. Computer 34 (11), 131–133. http://dx.doi.org/10.1109/2.963450.

Cohendet, P., Edward Steinmueller, W., 2000. The codification of knowledge: a conceptual and empirical exploration. Ind. Corp. Change 9 (2), 195–209.

Conboy, K., 2009. Agility from first principles: Reconstructing the concept of agility in information systems development. Inf. Syst. Res. 20 (3), 329–354. http://dx.doi.org/10.1287/isre.1090.0236.

Corbin, J.M., Strauss, A., 1990. Grounded theory research: Procedures, canons, and evaluative criteria. Qual. Sociol. 13 (1), 3–21. http://dx.doi.org/10.1007/BF00988593.

Corbin, J., Strauss, A., 2015. Basics of Qualitative Research: Techniques and Procedures for Developing Grounded Theory. Sage publications, California, United States, p. 431.

Cowan, R., Foray, D., 1997. The economics of codification and the diffusion of knowledge. Ind. Corp. Change 6 (3), 595–622. http://dx.doi.org/10.1093/icc/6.3.595.

Currall, S.C., Judge, T.A., 1995. Measuring trust between organizational boundary role persons. Organ. Behav. Human Decis. Process. 64 (2), 151–170.

Curtis, B., Herb, K., Neil, I., 1988. A field study of the software design process for large systems. Commun. ACM 31 (11), 1268–1287, http://doi.acm.org/10.1145/50087.50089.

Datta, P., Acar, W., 2010. Software and human agents in knowledge codification. Knowl. Manag. Res. Pract. 8 (1), 45–60. http://dx.doi.org/10.1057/kmrp.2009.34.

Dikert, K., Paasivaara, M., Lassenius, C., 2016. Challenges and success factors for large-scale agile transformations: A systematic literature review. J. Syst. Softw. 119, 87–108. http://dx.doi.org/10.1016/j.jss.2016.06.013.

Dingsøyr, T., Royrvik, E., 2003. An empirical study of an informal knowledge repository in a medium-sized software consulting company. In: 25th International Conference on Software Engineering, 2003. Proceedings. pp. 84–92. http://dx.doi.org/10.1109/ICSE.2003.1201190.

Dirks, K.T., Ferrin, D.L., 2001. The role of trust in organizational settings. Organ. Sci. 12 (4), 450–467. http://dx.doi.org/10.1287/orsc.12.4.450.10640.

Dorairaj, S., Noble, J., Malik, P., 2012. Knowledge management in distributed agile software development. In: 2012 Agile Conference. pp. 64–73. http://dx.doi.org/10.1109/Agile.2012.17.

Dybå, T., Dingsøyr, T., 2008. Empirical studies of agile software development: A systematic review. Inf. Softw. Technol. 50 (9–10), 833–859. http://dx.doi.org/10.1016/j.infsof.2008.01.006.

Edward Steinmueller, W., 2000. Will new information and communication technologies improve the 'codification' of knowledge? Ind. Corp. Change 9 (2), 361–376. http://dx.doi.org/10.1093/icc/9.2.361.

Ersoy, I.B., Mahdy, A.M., 2015. Agile knowledge sharing. Int. J. Softw. Eng. (IJSE) 6 (1), 1–15.

Flyvbjerg, B., 2006. Five misunderstandings about case-study research. Qual. Inq. 12 (2), 219–245. http://dx.doi.org/10.1177/1077800405284363.

Frigg, R., Hartmann, S., 2018. Models in science. In: Zalta, E.N. (Ed.), The Stanford Encyclopedia of Philosophy, Summer 2018 ed. Metaphysics Research Lab, Stanford University.

Ghobadi, S., Mathiassen, L., 2016. Perceived barriers to effective knowledge sharing in agile software teams. Inf. Syst. J. 26 (2), 95–125. http://dx.doi.org/10.1111/isj.12053.

Glazer, R., 1998. Measuring the knower: Towards a theory of knowledge equity. Calif. Manage. Rev. 40 (3), 175–194.

Gorschek, T., Gomes, A., Pettersson, A., Torkar, R., 2012. Introduction of a process maturity model for market-driven product management and requirements engineering. J. Softw.: Evol. Process 24 (1), 83–113. http://dx.doi.org/10.1002/smr.535.

Grant, R.M., 1996. Toward a knowledge-based theory of the firm. Strateg. Manag. J. 17 (S2), 109–122. http://dx.doi.org/10.1002/smj.4250171110.

Hall, M., 2006. Knowledge management and the limits of knowledge codification. J. Knowl. Manag. 10 (3), 117–126. http://dx.doi.org/10.1108/13673270610670894.

Hall, T., Beecham, S., Rainer, A., 2002. Requirements problems in twelve software companies: an empirical analysis. IEE Proc. - Softw. 149 (5), 153–160. http://dx.doi.org/10.1049/ip-sen:20020694.

Hendriks, P., 1999. Why share knowledge? The influence of ICT on the motivation for knowledge sharing. Knowl. Process Manag. 6 (2), 91–100. http://dx.doi.org/10.1002/(SICI)1099-1441(199906)6:2<91::AID-KPM54>3.0.CO;2-M.

Hislop, D., 2013. Knowledge Management in Organizations: A Critical Introduction. OUP Oxford.

Hoda, R., Noble, J., Marshall, S., 2012. Developing a grounded theory to explain the practices of self-organizing Agile teams. Empir. Softw. Eng. 17 (6), 609–639.

Izquierdo-Cortazar, D., Robles, G., Ortega, F., Gonzalez-Barahona, J.M., 2009. Using software archaeology to measure knowledge loss in software projects due to developer turnover. In: 2009 42nd Hawaii International Conference on System Sciences. pp. 1–10. http://dx.doi.org/10.1109/HICSS.2009.498.

Käpyaho, M., Kauppinen, M., 2015. Agile requirements engineering with prototyping: A case study. In: 2015 IEEE 23rd International Requirements Engineering Conference (RE). pp. 334–343. http://dx.doi.org/10.1109/RE.2015.7320450.

Karlsen, J.T., Hagman, L., Pedersen, T., 2011. Intra-project transfer of knowledge in information systems development firms. J. Syst. Inf. Technol. 13 (1), 66–80. http://dx.doi.org/10.1108/13287261111118359.

Karlsson, L., Dahlstedt, Å.G., Regnell, B., och Dag, J.N., Persson, A., 2007. Requirements engineering challenges in market-driven software development: An interview study with practitioners. Inf. Softw. Technol. 49 (6), 588–604. http://dx.doi.org/10.1016/j.infsof.2007.02.008, Qualitative Software Engineering Research.

Kaya, N., Patton, J., 2011. The effects of knowledge-based resources, market orientation and learning orientation on innovation performance: An empirical study of Turkish firms. J. Int. Dev. 23 (2), 204–219. http://dx.doi.org/10.1002/jid.1662.

Kittlaus, H.-B., Fricker, S., 2017. Software Product Management: The ISPMA-Compliant Study Guide and Handbook. Springer, http://dx.doi.org/10.1007/978-3-642-55140-6.

Kogut, B., Zander, U., 1992. Knowledge of the firm, combinative capabilities, and the replication of technology. Organ. Sci. 3 (3), 383–397. http://dx.doi.org/10.1287/orsc.3.3.383.

Kuusinen, K., Gregory, P., Sharp, H., Barroca, L., Taylor, K., Wood, L., 2017. Knowledge sharing in a large agile organisation: A survey study. In: International Conference on Agile Software Development. Springer, pp. 135–150.

Lai, L.F., 2007. A knowledge engineering approach to knowledge management. Inform. Sci. 177 (19), 4072–4094. http://dx.doi.org/10.1016/j.ins.2007.02.028.

Lavrakas, P., 2008. Encyclopedia of Survey Research Methods. http://dx.doi.org/10.4135/9781412963947.

Lenberg, P., Feldt, R., Wallgren, L.-G., 2014. Towards a behavioral software engineering. In: Proceedings of the 7th International Workshop on Cooperative and Human Aspects of Software Engineering. In: CHASE 2014, ACM, New York, NY, USA, pp. 48–55. http://dx.doi.org/10.1145/2593702.2593711.

Lenberg, P., Feldt, R., Wallgren, L.-G., 2015. Behavioral software engineering: A definition and systematic literature review. J. Syst. Softw. 107, 15–37. http://dx.doi.org/10.1016/j.jss.2015.04.084.

Li, J., Moe, N.B., Dybå, T., 2010. 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. ESEM '10, ACM, New York, NY, USA, pp. 13:1–13:10. http://dx.doi.org/10.1145/1852786.1852804.

MacCormack, A., Verganti, R., 2003. Managing the sources of uncertainty: Matching process and context in software development. J. Prod. Innov. Manage. 20 (3), 217–232. http://dx.doi.org/10.1111/1540-5885.2003004.

McChesney, I.R., Gallagher, S., 2004. Communication and co-ordination practices in software engineering projects. Inf. Softw. Technol. 46 (7), 473–489. http://dx.doi.org/10.1016/j.infsof.2003.10.001.

Melnik, G., Maurer, F., 2004. Direct verbal communication as a catalyst of agile knowledge sharing. In: Proceedings of the Agile Development Conference, ADC 2004. pp. 21–31. http://dx.doi.org/10.1109/ADEVC.2004.12.

Melo, C.d.O., Cruzes, D.S., Kon, F., Conradi, R., 2013. Interpretative case studies on agile team productivity and management. Inf. Softw. Technol. 55 (2), 412–427. http://dx.doi.org/10.1016/j.infsof.2012.09.004, Special Section: Component-Based Software Engineering (CBSE), 2011.

Miller, D., Shamsie, J., 1996. The resource-based view of the firm in two environments: The hollywood firm studios from 1936–1965. Acad. Manag. J. 39 (3), 519–543. http://dx.doi.org/10.2307/256654.

Nieves, J., Quintana, A., Osorio, J., 2014. Knowledge-based resources and innovation in the hotel industry. Int. J. Hosp. Manag. 38, 65–73. http://dx.doi.org/10.1016/j.ijhm.2014.01.001, cited By 36.

Nonaka, I., 1994. A dynamic theory of organizational knowledge creation. Organ. Sci. 5 (1), 14–37. http://dx.doi.org/10.1287/orsc.5.1.14.

Nonaka, I., Takeuchi, H., Umemoto, K., 1996. A theory of organizational knowledge creation. Int. J. Technol. Manage. 11 (7–8), 833–845.

Nonaka, I., Toyama, R., Konno, N., 2000. SECI, ba and leadership: a unified model of dynamic knowledge creation. Long Range Plan. 33 (1), 5–34. http://dx.doi.org/10.1016/S0024-6301(99)00115-6.

OECD, 2022. Entrepreneurship - enterprises by business size - OECD data. http://dx.doi.org/10.1787/31d5eeaf-en.

Ouriques, R., Wnuk, K., Berntsson Svensson, R., Gorschek, T., 2018. Thinking strategically about knowledge management in agile software development. In: Kuhrmann, M., Schneider, K., Pfahl, D., Amasaki, S., Ciolkowski, M., Hebig, R., Tell, P., Klünder, J., Küpper, S. (Eds.), Product-Focused Software Process Improvement. Springer International Publishing, Cham, pp. 389–395.

Ouriques, R., Wnuk, K., Svensson, R.B., Gorschek, T., 2019. Knowledge management strategies and processes in agile software development: A systematic literature review. Int. J. Softw. Eng. Knowl. Eng. 29 (3), 345–380. http://dx.doi.org/10.1142/S0218194019500153.

Paredes, J., Anslow, C., Maurer, F., 2014. Information visualization for agile software development. In: 2014 Second IEEE Working Conference on Software Visualization. pp. 157–166. http://dx.doi.org/10.1109/VISSOFT.2014.32.

Rus, I., Lindvall, M., 2002. Knowledge management in software engineering. IEEE Softw. 19 (3), 26–38. http://dx.doi.org/10.1109/MS.2002.1003450.

Ryan, S., O'Connor, R.V., 2013. Acquiring and sharing tacit knowledge in software development teams: An empirical study. Inf. Softw. Technol. 55 (9), 1614–1624. http://dx.doi.org/10.1016/j.infsof.2013.02.013.

Saito, S., Iimura, Y., Massey, A.K., Antón, A.I., 2018. Discovering undocumented knowledge through visualization of agile software development activities. Requir. Eng. 23 (3), 381–399. http://dx.doi.org/10.1007/s00766-018-0291-4.

Santos, V., Goldman, A., De Souza, C.R., 2015. Fostering effective inter-team knowledge sharing in agile software development. Empir. Softw. Eng. 20 (4), 1006–1051. http://dx.doi.org/10.1007/s10664-014-9307-y.

Seaman, C., 1999. Qualitative methods in empirical studies of software engineering. IEEE Trans. Softw. Eng. 25 (4), 557–572. http://dx.doi.org/10.1109/32.799955.

Sedano, T., Ralph, P., Péraire, C., 2017. Software development waste. In: Proceedings of the 39th International Conference on Software Engineering. ICSE '17, IEEE Press, Piscataway, NJ, USA, pp. 130–140. http://dx.doi.org/10.1109/ICSE.2017.20.

Sirmon, D., Hitt, M., 2009. Contingencies within dynamic managerial capabilities: Interdependent effects of resource investment and deployment on firm performance. Strateg. Manag. J. 30 (13), 1375–1394. http://dx.doi.org/10.1002/smj.791, cited By 159.

Soliman, F., Spooner, K., 2000. Strategies for implementing knowledge management: role of human resources management. J. Knowl. Manag. 4 (4), 337–345. http://dx.doi.org/10.1108/13673270010379894.

Soomro, A.B., Salleh, N., Mendes, E., Grundy, J., Burch, G., Nordin, A., 2016. The effect of software engineers' personality traits on team climate and performance: A systematic literature review. Inf. Softw. Technol. 73, 52–65. http://dx.doi.org/10.1016/j.infsof.2016.01.006.

Sørensen, C., Lundh-Snis, U., 2001. Innovation through knowledge codification. J. Inf. Technol. 16 (2), 83–97. http://dx.doi.org/10.1080/713772762.

Steen, O., 2007. Practical knowledge and its importance for software product quality. Inf. Softw. Technol. 49 (6), 625–636.

Stettina, C.J., Heijstek, W., Fægri, T.E., 2012. Documentation work in agile teams: The role of documentation formalism in achieving a sustainable practice. In: 2012 Agile Conference. pp. 31–40. http://dx.doi.org/10.1109/Agile.2012.7.

Stol, K., Ralph, P., Fitzgerald, B., 2016. Grounded theory in software engineering research: A critical review and guidelines. In: 2016 IEEE/ACM 38th International Conference on Software Engineering. ICSE, pp. 120–131. http://dx.doi.org/10.1145/2884781.2884833.

Strode, D.E., Huff, S.L., Hope, B., Link, S., 2012. Coordination in co-located agile software development projects. J. Syst. Softw. 85 (6), 1222–1238. http://dx.doi.org/10.1016/j.jss.2012.02.017.

Swedish Research Council, 2017. Good Research Practice. Swedish Research Council, Stockholm, Sweden, p. 86.

Szulanski, G., Cappetta, R., Jensen, R.J., 2004. When and how trustworthiness matters: Knowledge transfer and the moderating effect of causal ambiguity. Organ. Sci. 15, 600–613.

Tom, E., Aurum, A., Vidgen, R., 2013. An exploration of technical debt. J. Syst. Softw. 86 (6), 1498–1516. http://dx.doi.org/10.1016/j.jss.2012.12.052.

West, D., Grant, T., Gerush, M., D'Silva, D., 2010. Agile development: Mainstream adoption has changed agility. Forrester Res. 2 (1), 41.

Williams, L., Cockburn, A., 2003. Agile software development: it's about feedback and change. Computer 36 (6), 39–43. http://dx.doi.org/10.1109/MC.2003.1204373.

Wohlin, C., Runeson, P., Höst, M., Ohlsson, M.C., Regnell, B., Wesslén, A., 2012. Experimentation in Software Engineering. Springer Science & Business Media.

Raquel Ouriques is a Ph.D. student in Software Engineering at Blekinge Institute of Technology. Her research interest includes developing studies/strategies to improve software-intensive product development organisations processes, applying management practices that consider knowledge as a key to combine strategy and technology.

Krzysztof Wnuk is an associate professor at the Software Engineering Research Group (SERL) of Blekinge Institute of Technology, Sweden. He received his M.Sc. Degree from Gdansk University of Technology, Poland (2006) and his Ph.D. from Lund University, Sweden (2012). His research interests include market-driven software development, requirements engineering, software product management, decision making in requirements engineering, large-scale software, system and requirements engineering and management and empirical research methods. He is interested in software business, open innovation and open source software. He works as an expert consultant in software engineering for the Swedish software industry.

Tony Gorschek is a Professor of Software Engineering at Blekinge Institute of Technology — where he works as a research leader and scientist in close collaboration with industrial partners. Dr. Gorschek has over fifteen years industrial experience as a CTO, senior executive consultant and engineer. In addition he is a serial entrepreneur — with five startups in fields ranging from logistics to internet based services and database register optimisation. At present he works as a research leader and in several research projects developing scalable, efficient and effective solutions in the areas of Requirements Engineering, Product Management, Value based product development, and Real Agile™and Lean product development andevolution. Dr. Gorschek leads the SERT profile (Software Engineering ReThought) developing the next generation of applied empirical research movements to meet the challenges of the next generation of software intensive products and services. www.rethought.se www.gorschek.com

Dr. Richard Berntsson Svensson is an Associate Professor in Software Engineering at Chalmers | university of Gothenburg, Sweden. His research interests include data-driven decision making, agile and lean software development, requirements engineering, creativity and innovation, and human aspects of software engineering. He received his Ph.D. from Lund University, Sweden, 2011