Knowledge Activities applied — Towards a Holistic Knowledge Management Approach in the Software Industry

Martin Dietze
Dermalog Identification Systems, Hamburg, Germany
martin.dietze@dermalog.com

Marion Kahrens
European College of Business and Management, London, UK
mkahrens@eurocollege.org.uk

Abstract

Abstract

Purpose — This paper aims to close the gap between the generic concept of knowledge activities and implementing them in the context of software engineering organisations concentrating on the non-technical aspects, like team organisation and practices.

Design/methodology/approach — This qualitative research used a questionnaire with practitioners such as software developers and team leads who were asked to provide feedback on a set of team practices and measures typically used in software engineering projects and assess their relation to the activities of acquiring, codifying, storing, maintaining, transferring and creating knowledge. The obtained results were analysed using frequency analysis and further descriptive statistics yielding a matrix linking the investigated team practices and measures to knowledge activities.

Findings — Team practices and measures commonly applied in software engineering can be facilitated to trigger particular knowledge activities. While most of these team practices and measures originate from agile methods they are not restricted to these. A purposeful composition can help in assembling a balanced set of knowledge activities aimed at fostering given knowledge goals in software engineering organisations.

Originality/value — This is the first study investigating application and relevance of Knowledge Activities in the software industry by linking them to practices and measures well-accepted in software engineering, thus providing the necessary vocabulary for the implementation of knowledge management in software development teams.

Practical implications — By bridging the communication and terminology gap between knowledge management research and software engineering practitioners, this work lays the foundation for assessing software teams' knowledge profiles more easily and creating prerequisites for implementing knowledge management by facilitating common practices and measures often already part of their daily work. Hence overhead can be avoided when implementing knowledge management.

Keywords Knowledge Creation, Knowledge Activities, SECI Model, Software Engineering, Development Team Practices

Paper type Research paper

1 Introduction

The software industry had to find ways of dealing with knowledge at a time when knowledge management as a scientific discipline had not yet evolved since software development is simply inherently knowledge-intensive. The emergence of *agile* approaches to planning, executing and managing software

projects (Abrahamsson *et al.* 2002) was an important step forward, not only because of its success (Mansoor *et al.* 2019) but also from a knowledge management perspective with its emphasis on knowledge transfer, collaboration and face-to-face communication. In the 1990s, after works like Nonaka's SECI model (Nonaka and Takeuchi, 1995) and first notions of knowledge activities (Pentland, 1995; Andersen, 1996) had been published, software developers' best practice started being complemented by research addressing the particularities of knowledge, knowledge management and creation in software engineering. Some researchers focused on support by tools like e.g. Wikis and issue trackers to support knowledge transfer, maintenance and storage (Dingsøyr and Conradi, 2002; C. Khalil and S. Khalil, 2019; Sungkur and Ramasawmy, 2014; Abdur *et al.* 2015). Others, like Dissanayake *et al.* (2013), revealed the correspondence of elements in agile software development methodologies to key knowledge management concepts as described by Nonaka and Takeuchi (1995), Nonaka and Konno (1998) and Nonaka and von Krogh (2009).

However, while the combination of knowledge management and software engineering as a field of interest for researchers has now been around for at least two decades, a separation between the two camps is still noticeable and calls for a more holistic approach trying to connect knowledge management to existing elements of software development teams' work. This research, an extension to the conference paper presented at the IFKAD 2021 in Rome (Dietze and Kahrens, 2021), is the very first application of the concept of knowledge activities as proposed by Heavin and Adam (2012) in the field of software development and bridges this communication and terminology gap between knowledge management research and software engineering practitioners by investigating a selection of software development team practices and measures, assessing their impact on team knowledge and establishing their relation to six knowledge activities. Having connected the concept of knowledge activities to practitioners' daily work now also their relevance and their already existing application in software teams are revealed. With this understanding at hand, software project leaders are able to more consciously involve the concept of knowledge when designing team processes.

The paper is organised as follows: In section 2 relevant existing work is reviewed, followed by two research questions. The research method used in this study is outlined in section 3. The results are presented in section 4 followed by the conclusion in section 5.

2 Literature Review

2.1 Software Engineering

Software Engineering is the systematic application of engineering approaches to the development of software (Laplante, 2007). Hence, it goes far beyond the writing of program code as it covers the practical problems of producing software (Sommerville, 2008) and the project lifecycle from planning to completion.

The classic approach to software projects is the waterfall model originating in the manufacturing and construction industries and breaking down the project into a sequence of phases, each depending on its predecessor (Benington, 1983). The Project Management Institute (2013) defines five groups of project activities during the project life cycle. These activity groups are project initiation, planning, executing, monitoring and controlling and closing. Alecu (2011) emphasises that these major project activity groups are seen as industry-independent and they should not be mixed up with the project life-cycle phases that are the following: concept or initial idea, definition of the preferred solution, development of the solution, handover or delivery of the software product and formal closure (Project Management Institute, 2013). Bryde *et al.* (2018) propose active project management activity to capture and apply the knowledge related to all relevant project activities during the whole project lifecycle rather than just in retrospective.

Over the years the family of agile methodologies have gained more and more importance. The Agile Manifesto lists four principles: Individuals and interactions over processes and tools, working software over comprehensive documentation, customer collaboration over contract negotiation, responding to change over following a plan (Beck *et al.* 2001). Agile methodologies excel in their ability to create business value while still embracing requirement changes as well as better dealing with complexity and uncertainty (Little, 2005; Jeon *et al.* 2011; Maiti and Mitropoulos, 2015). They have established

themselves and replaced the waterfall model in vast parts of the software industry (Rodríguez *et al.* 2012; Dima and Maassen, 2018). Maiti and Mitropoulos (2015) note that traditional agile methodologies usually focus on functional requirements (i.e. what the end user sees) at the cost of the non functional (infrastructure, technical support features). This can pose a risk, as efforts required to finish promised functionality can be overlooked (Behutiye *et al.* 2019). Over the years various flavours of agility have been proposed, all of them sharing the set of values from the manifesto, yet exhibiting differences in the way they are executed. Abrahamsson *et al.* (2002) list ten classic agile frameworks of which three will be briefly introduced here: Feature Driven Development (FDD) for being probably the best compromise between agile and waterfall, Extreme Programming (XP) for its historic significance and the terminology originating from it and Scrum as the most popular agile framework today.

Introduced in 1999, FDD is built around a feature list consisting of compact descriptions for all that needs to be implemented (Palmer and Felsing, 2001). In this way the feature list is for FDD what a scope of work is for traditional planning, and it requires a full analysis of the future system. The overall system's implementation effort is calculated from all the individual features' estimations which is — due to each feature's limited complexity — relatively reliable. This approach makes FDD more compatible with traditional planning than other agile methodologies. XP was proposed in 1999 and stresses a number of concepts deemed desirable for software development: focusing on the development process, short planning cycles, teamwork and communication (Beck, 1999). Features are described in so-called user stories, usually implemented in pair programming (Juric, 2000) and released in short iterations. Scrum was already developed in the mid-nineties and became better-known in the software community along with the Agile Manifesto in 2001 Schwaber and Beedle (2001). The term 'Scrum' was borrowed from the Rugby sport where it denotes a bunch of players. This was inspired by the paper "The New New Product Development Game" (Takeuchi and Nonaka, 1986) where the authors describe a very team-oriented, holistic ('Rugby') approach to development. Scrum relies on a very simple rule set, a workflow including short iterations (sprints) and a set of various team meetings held on a regular basis (Sutherland and Schwaber, 2010). Nowadays Scrum is the most widely adopted agile framework, and more recent incarnations of the idea are gaining importance in the software industry.

	Scrum	Feature-Driven Development
Agility	more	less
Centralisation	less	more
Predictability	lower	higher
Strengths	quality, embracing changes	planning accuracy, cost estimation
Applicability	small, medium projects	large projects

Table I. Scrum vs. FDD according to Umbreen *et al.* (2015)

The Modern Agile initiative is a relatively new movement trying overcome shortcomings often observed in today's agile projects and instead return to the roots of what was originally considered 'agile', proposing an iteration to the original agile principles: Make people awesome, make safety a prerequisite, experiment & learn rapidly, deliver value continuously (*Modern agile web site* n.d.; Mansoor *et al.* 2019). While Modern Agile aims to be *more* agile, an agile derivative moving into the opposite direction is getting more and more relevant, in particular for large projects. Hybrid Agile tries to benefit from agile elements in traditional plan-driven projects (Agile Alliance *et al.* n.d.). Because time to market and planable costs tend to get more and more critical to customers with growing project scale, pure agile methodologies fail to qualify while some of their productivity, and flexibility can nevertheless be facilitated with the hybrid approach (Raval and Rathod, 2014; Imani, 2017). It is worthwhile noting that psychological safety, the second of Modern Agile's guiding principles, is also one of the enablers of knowledge creation (von Krogh, Ichijo *et al.* 2000).

2.2 Knowledge Creation

The fundamental concepts of what is known today as Knowledge Management and Knowledge Creation have been created and further developed since the 1990s, and a vast amount of relevant research on how to implement KM in — usually large scale — enterprises exists (Nonaka and Takeuchi, 1995; von Krogh, Ichijo *et al.* 2000; Ichijo and Nonaka, 2008). Nonaka and Takeuchi (1995) define knowledge as 'justified true belief' held by individuals. In order to qualify as knowledge rather than just information a "clear understanding of information and their associated patterns" is an additional important requirement (Bierly *et al.* 2000, p. 600). Knowledge is also the actuality of skilful action and/or the potentiality of creating situations to permit (skilful) action (Nonaka and von Krogh, 2009).

Nonaka and Takeuchi (1995) point out that there is a distinction between explicit and tacit knowledge. Tacit knowledge comprises information that is difficult to express, formalise or share. It resides within people and according to Stenmark (2000) it can neither be documented in a manual nor be explained in words to others. Hence this kind of knowledge requires involving the knowing one and transferring the knowledge via periods of practices where experiences are shared through actions. In other words, knowledge needs to be transmitted from person to person (Grover and Davenport, 2017).

In opposition to tacit knowledge, explicit knowledge can be transmitted by using strings, i.e. interaction between physical objects, in the right circumstances (Collins, 2010). It can be codified and/or verbalised (Nonaka and Toyama, 2003; Grover and Davenport, 2017) and expressed by data numbers, formulas, pictures, manuals, patents (Holste and Fields, 2010). Both explicit and tacit knowledge can be passed on and converted using appropriate methods and with varying degrees of difficulty or required effort. The resulting four possible transitions — tacit-to-tacit (Socialisation), tacit-to-explicit (Externalisation), explicit-to-explicit (Combination) and explicit-to-tacit (Internalisation) are the pillars of the SECI model, according to which organisational knowledge creation evolves from a spiral out of continuously transforming and transferring explicit and tacit knowledge between individuals (Nonaka and Takeuchi, 1995).

Since the beginning of the SECI model's development, a broad academic debate has arisen, mainly around the distinctions between the different conversion processes, the relationship between the explicit and tacit levels and its possibilities related to cultural differences. The adoption and application of the SECI model is under continuous consideration and development (von Krogh, Nonaka *et al.* 2012; C. S. Lee and Kelkar, 2013; Nezafati *et al.* 2009; Tee and S. S. Lee, 2013). The SECI model is widely accepted but varying contents and perceptions regarding the importance of particular aspects of the knowledge creation model exist, such as cultural aspects, the practical implications of the transformation of knowledge and the role of management (Kahrens and Früauff, 2018). An important extension, the concept of Ba (Japanese for 'space'), introduced by Nonaka and Konno (1998), stating that in order for the SECI model's transitions to take place an enabling context is necessary. Hence the four pillars are complemented by four specific types of Ba.

In addition, further enablers of knowledge have been proposed: psychological safety and care (von Krogh, Ichijo *et al.* 2000), micro-communities of small to medium size and changing membership over the years, and sources of innovation like e.g. R&D departments (Ichijo and Nonaka, 2008) or, like in tech companies (e.g. Intel), innovation-driving development and production (Teece, 2010). Similar principles can be applied to dealing with peers outside the organisation, like building communities with customers in order to share knowledge through co-experience and shared praxis (Sapir *et al.* 2016; von Krogh, Ichijo *et al.* 2000; Ichijo and Nonaka, 2008). Saifi *et al.* (2016) propose to encourage initiative and participation inside an organisation whereby decentralised decision-making and reduced formalisation are preferred as they encourage initiative and participation. While the SECI model starts from the point where individuals already carry knowledge to enter the knowledge creation spiral, the idea of distinguishing between embodied and not-yet embodied ('self-transcending') tacit knowledge is introduced in Scharmer (2001). Scharmer addresses the question for the driving force behind the knowledge spiral, stressing the importance of appropriate enabling contexts and the necessary conversational complexity in communication. In the future, the application and extension of the SECI model are expected to be leaner in structures and more digitised concerning knowledge creation and sharing (Kahrens and Früauff, 2018).

The SECI model has also been subject to criticism, a.o. for being too vague in its recommendations and its perception of both, tacit and explicit knowledge (Schreyögg and Geiger, 2003; Schreyögg and Geiger, 2004; Lin, 2021), also for being deeply rooted in Japanese culture and hence not sufficiently considering cultural differences as possible obstacles to its application (Andreeva and Ikhilchik, 2011). Nevertheless, this research will consider the basic concept of the SECI model when assessing typical meeting structures in software development projects to improve the exchange among team members and thereby increase learning and knowledge.

2.3 Knowledge Management in Software Development

It is a common understanding that software engineering and development is a highly knowledge-intensive discipline, and various aspects of it have attracted the interest of researchers in knowledge management (Rus and Lindvall, 2002; Kavitha and Irfan, 2011; Vasanthapriyan *et al.* 2015; Shongwe, 2017). In terms of the taxonomy proposed by Earl (2001), knowledge management strategies are usually closely related to the technocratic schools for traditional software development and more to the behavioural schools for agile approaches (Bjørnson and Dingsøyr, 2008). Hegde and Walia (2014) note that generally the different involved forms and dimensions of knowledge have a significant impact on the developers' creativity.

An early study based on 'lessons learnt' reports from eight software companies (Dingsøyr and Conradi, 2002) shows that employees are interested in this field and also willing to invest in it, e.g. by developing supporting tools. From analysing knowledge flows involved when creating source code Sandhawalia and Dalcher (2010) conclude that the activities of knowledge creation, learning and reflection clearly play an important role in uncovering and reviewing mistakes and mismatches. Similarly, Ward and Aurum (2004) find that knowledge management helps appreciate the challenges and complexities inherent in software development while also revealing that employees found the helpers and techniques they were using inadequate to support the intended knowledge management.

A significant number of studies stresses the importance of software tool support. The use of knowledge repositories (e.g. Wikis, databases etc.) is considered a central element of knowledge management in software organisations (Dingsøyr and Conradi, 2002; C. Khalil and S. Khalil, 2019). However, this term is a bit misleading as persisted knowledge becomes *information* and needs to be transferred back to *knowledge* when consumed by others. Bjørnson and Dingsøyr (2008) confirm a tendency to over-emphasise the role of technology by finding that some software engineering organisations focus on storage and retrieval of knowledge at the cost of knowledge transfer, application and creation. However, collaborative software tools like issue trackers and Wikis are considered useful (Sungkur and Ramasawmy, 2014; Abdur *et al.* 2015).

Dissanayake *et al.* (2013) point out that agile methodologies with their team processes already contain various elements of what is known from the SECI model and Ba. Similarly, Abdur *et al.* (2015) find processes and tools common in agile beneficial for knowledge creation. These findings are confirmed by an investigation in large French companies conducted by C. Khalil and S. Khalil (2019) who conclude that agile practices contribute to the creation of a knowledge-intensive culture. Agile methodologies' focus on team dynamics and information flow are beneficial for competitiveness more or less come automatically in small or medium-sized enterprises (Basri and O'Connor, 2011; Shongwe, 2017; Heavin and Adam, 2014). But also larger organisations can benefit from it since agile teams are usually kept small on purpose: e.g. Scrum recommends teams of three to nine people (Sutherland and Schwaber, 2010).

Elements fostering open discussion or collaboration have a strong influence on team knowledge. One is the Retrospective in Scrum where team members gather after each sprint reflecting on what did not go well in the finished iteration and which measures should be taken to prevent it from happening again (Viana *et al.* 2013). Similarly in projects based on the waterfall model lessons learnt meetings take place after milestones. Another example is Pair Programming where two developers share the same screen while composing source code together (Kavitha and Irfan, 2011).

2.4 Knowledge Activities

While knowledge clearly plays a crucial role in software engineering, still some measures are required in order to achieve or design a systematic process of knowledge creation. For this, researchers propose the definition of Knowledge Activities (KA) — transactions or manipulations of knowledge where the knowledge is the object, not the result (Jetter *et al.* 2006; Kraaijenbrink *et al.* 2006). The term Knowledge Activity is fairly generic, hence several different interpretations are in use. Table II provides a selection of variations to the concept.

Authors	Name	Activities
Holsapple and Joshi (2004)	Knowledge Manipulation Activities	Acquiring, Selecting, Internalising, Using, Externalising, Generating
Ward and Aurum (2004)	KM Activities	Application, Distribution, Organisation, Adaptation, Identification, Acquisition, Creation
Costa and Monteiro (2017)	Knowledge Management Processes	Acquisition, Storage, Codification, Sharing, Application, Creation
Kraaijenbrink *et al.* (2006)	Knowledge Activities	Elicitation, Codification, Detection, Assessment, Transfer (of knowledge), Transfer (of knowledge holder), Nurturing, Motivation
Heavin and Adam (2012)	Knowledge Activities	Acquire, Codify, Store, Maintain, Transfer, Create

Table II. Variants to the concept of Knowledge Activities

Heavin and Adam (2012) derive a list of six Knowledge Activities — acquire, codify, store, maintain, transfer, create — from the definition provided by Holsapple and Joshi (2004) in an investigation of KA application in five Irish software SMEs finding a dominance of knowledge and an underrepresentation of knowledge creation (Heavin and Adam, 2012; Heavin and Adam, 2014). Ward and Aurum (2004) focus on seven separate activities — knowledge application, distribution, organisation, adaptation, identification, acquisition and creation — finding them used and evolving with technology, organisational culture and practises. In a subsequent study, the same group of researchers investigate how and whether all these activities are executed by looking at four projects in two software companies and linking them to typical project lifecycle stages: Requirements Gathering, Analysis & Design, Building & Development, Testing and Deployment (Aurum *et al.* 2008). Their findings indicate that while the KAs are executed, this often happens implicitly, i.e. without having been identified as KAs by developers. Although there clearly is a common understanding of the concept behind knowledge activities, there are differences in detail e.g. Costa and Monteiro (2017) refer to knowledge sharing and application instead of transferring and maintaining.

Hence it can be concluded that the concept of knowledge activities (or KM activities or knowledge management processes) is recognised as a relevant element in implementing knowledge management. To a far lesser extent, the question has been addressed which software development practices actually result in executing which KAs. Sungkur and Ramasawmy (2014) refer to such activities as building blocks for knowledge management processes to which technical tools are linked in order to show which aspects of knowledge management they support. In their survey on knowledge management in software engineering, Vasanthapriyan *et al.* (2015) go through the software development life cycle and analyse its stages for their relation to the teams' knowledge, while employees from small South African software companies comment on their experience regarding KAs in Shongwe (2017). Examples for such mappings, while not actually focused on, can be found in some other works, like Sandhawalia and Dalcher (2010) and Abdur *et al.* (2015).

Mapping of some team processes and measures to knowledge activities can be found as side products in some of the abovementioned studies, like e.g. Pair Programming, Daily Scrum, rotation/visit in Abdur *et al.* (2015), or design, test, implementation, document, development in Aurum *et al.* (2008). However, a systematic analysis of how elements of software development workflows and management measures relate to knowledge activities would be of value to both researchers and practitioners, which is why this research is dedicated to this aspect.

2.5 Research Questions

The purpose of this study was to develop a holistic concept of implementing knowledge activities in the context of software engineering organisations.

Therefore, the overarching research questions for this qualitative exploratory study were:

RQ1. Which knowledge activities are related to which team practices and measures, how are these practices and measures in use in software engineering organisations and assessed from the knowledge perspective?

RQ2. How should team practices and measures be combined in order to support knowledge activities suitable for effective knowledge management in software engineering organisations?

3 Research Method

3.1 Qualitative Research

In accordance with the proposed research questions, this study used a qualitative research approach. Although such a kind of qualitative research allows for observing a phenomenon in-depth (Eisenhardt and Graebner, 2007) and case study research would seem to be appropriate, the authors of this paper decided to take a broader view that is not bound to a single case organisation.

In contrast, quantitative research would assume the researchers to be conscious of all problems involved and the variables' relationship in interpreting their correlations (Silverman, 2014). This was not the case for this research because in terms of complexity, challenges in software development and delivery are related to the elements of software development projects consisting of non-trivial interactions among project members internally and externally. Qualitative research methods are more helpful to analyse abstract constructs of observable phenomenon (Creswell, 2014).

This study analysed how knowledge activities and their structured and purposeful implementation benefit software engineering organisations. The underlying assumptions were based on a partially phenomenological stance. According to van Manen (2007), a context-sensitive form of interpretive inquiry, the phenomenology of practice, is well suited to serve practitioners who in their day-to-day practice may be unaware of the depths of people's experiences. By constructing the realities related to knowledge activities in developing and programming software solutions this qualitative empirical method composed summaries of lived experience and asked a group of experts to assess these practices. Thereby, the underlying patterns and structures of meaning can be drawn when a central feature of the phenomenon is considered (van Manen, 2016). The interpretive findings of this research serve as the foundation for future empirically conducted research aiming to measure the degree to which knowledge activities applied in software engineering projects or other service industries contribute to the organisational knowledge base.

3.2 Sampling and data collection

Several factors contribute to the success of knowledge activities within organisations including the firm's objectives, the enabling conditions and the environment. The implication here is that knowledge activities are best examined within the context where they take place. A key advantage of qualitative research is that the researchers are free to choose from multiple sources while still incorporating one's know-how to extract understanding.

In this research, data was collected through questionnaires. The predominant closed questions were self-constructed but developed in accordance with the knowledge activities' definition provided

by Heavin and Adam (2012) and extended or rather adjusted for the industrial focus on software engineering based on the authors' experience as researchers and practitioners as well as the knowledge activities' theoretic concept. For each question, frequency analysis was used to analyse the quantitative data and performed using MS Excel. Due to the fact that the collected data was of quantitative nature, the coding took place a priori by transforming e.g. the 6-point Likert scales into numbers (1-6). The participants were project managers, software architects, developers, and other experts of software development.

The sampling strategy for the data collection followed the principle of convenient purposive sampling that requires recruiting from a group or organisations with a close perspective on the research phenomenon of interest (Creswell, 2014). The convenient purposive sampling took place by posting the questionnaire on online research platforms (SurveyCircle) and social media (LinkedIn) approaching special expert groups related to software project management or agile methodologies. Although convenience sampling strategies do not allow generalising the findings due to the unknown population, it plays a prominent role in the field of business and management (Bryman and Bell, 2007). The expectation was to reach 50 participants with a self-administrated internet-mediated questionnaire.

It was not the purpose of this qualitative research to achieve representativeness of the results. Entering into the network of the researchers and identifying respondents allowed targeted sampling to ensure the sample to contain enough variations along demographic and organisational dimensions for drawing conclusions. While suitable sample sizes for heterogeneous populations in qualitative research are recommended as being between 12 - 30 respondents, the authors of this paper decided to attract more participants in order to ensure the comparison between distinct groups of them. Table III illustrates the composition of the participants.

Role	Team Lead	Developer	Architect	Other	
	40.48%	30.95%	5.95%	22.62%	
Organisation employees	1 ... 10	11 ... 50	51 ... 250	251 ... 500	over 500
	10.71%	7.14%	30.36%	12.50%	39.29%
Organisation core business	Standard Software	Individual Software	Products incl. software	Services incl. software	Others
	3.57%	19.64%	26.79%	26.79%	23.21%

Table III. Composition of the participants

RQ1 and RQ2 guided the review of the questionnaire data which was analysed by frequency analysis statements related to typical software development team practices and measures. Consistency of the questionnaire was proven in advance by pretesting it two times with software development experts in order to avoid respondents interpreting the questions differently. Thereby, the questionnaire's robustness and reliability were approved so that consistent findings were produced.

4 Results and Interpretation

4.1 Various companies' approaches to software development

Responses were obtained from practitioners working for companies of 5 different scales. The vast majority of these companies develop individual software or software as part of a service or a product. The majority uses some mixture of software development methodologies while a significantly smaller number focuses on a single dominant one (more than 75%). How software development methodologies were executed in the respondents' companies is shown in figure 1.

See external file: *methodologies.eps*

The outcome revealed a high proportion of hybrid and other agile methodologies (55%). This may seem surprising, however when looking at the results obtained from small (up to 50 employees) and

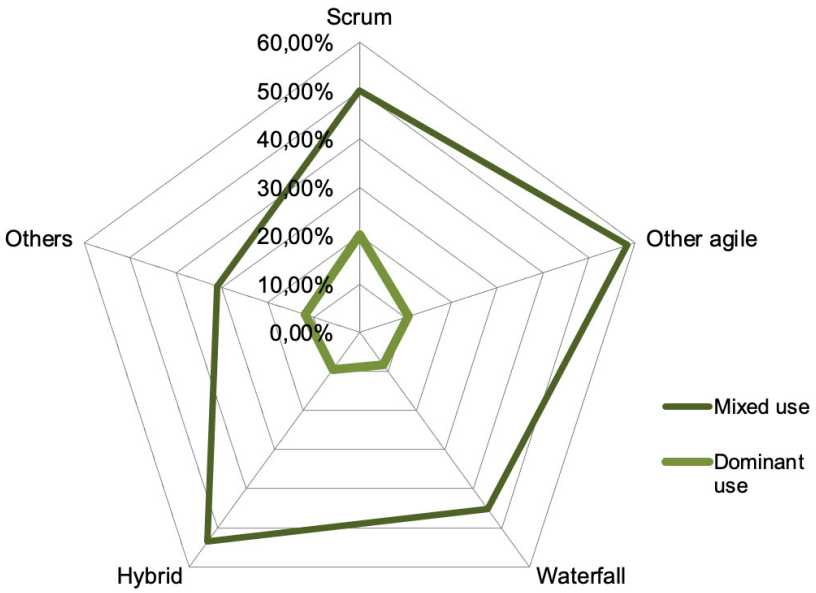

Figure 1. Mix of Software Development Methodologies

larger (more than 50 employees) companies separately, these two groups exhibit different preferences:

- In the 'small' subgroup a significantly higher number of responses indicated the dominant rather than mixed use of some methodology with Scrum in the lead (37.5%) followed by waterfall, hybrid and others (each 12.5%).
- In the 'larger' subgroup methodologies tend to be used in combination. Here hybrid (64.9%) is used most frequently, followed by other agile (56.8%) and Scrum (50%).

This finding is consistent with the fact of small companies being more easily able to adopt radical changes. Agile software development methodologies are still relatively young, and introducing them for dominant use has an impact not only on development teams but also on management structures and even customers. Another aspect is project scale as shown in section 2.1. Scrum has its main strengths in smaller projects while other methodologies perform well in larger projects. Assuming that large companies implement larger projects than small companies, their respective choices of methodologies found in the questionnaire support this.

Both, the overall dominance of Scrum, hybrid and other agile indicate that at least agile elements have made their way into the vast majority of companies. Given the fact that most of the investigated practices and measures originate from agile methodologies, the respondents' familiarity with them can now be safely assumed.

4.2 Knowledge Activities

The knowledge activities' relevance was recognised by a vast majority of the respondents while their respective companies' efficiency in addressing was assessed rather critically (figure 2).

See external file: *kas.eps*

With 94.4% knowledge acquisition is the most recognised activity. This is plausible since the effect of new staff or training courses is very obvious to employees. Conversely knowledge codification (83.3%) is deemed least relevant. This may indicate lack of experience with some respondents due to a possible lack of emphasis on this in their respective organisations. When it comes to efficiency, a majority sees serious deficits in updating and creating knowledge. Looking at subgroups some interesting differences can be found:

- Employees from companies dominantly using agile methodologies generally consider themselves

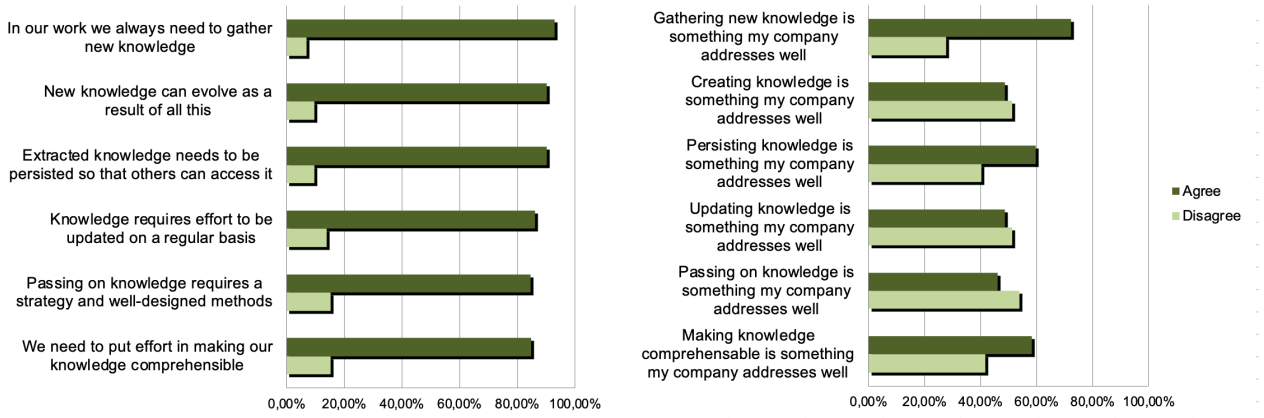

Figure 2. Knowledge Activities' Relevance and Efficiency

more proficient in executing knowledge activities than their counterparts from companies preferring traditional methodologies.

- The same applies for employees from companies categorised as 'small' above in comparison those from 'larger' organisations.

This similarity is not surprising since agile methods were found to be dominating in small organisations in section 4.1.

4.3 Team Practices and Measures

With only two exceptions (cross-disciplinary task forces and rotating of team responsibilities) the chosen team practices and measures were assessed positively with an agreement of 60% or more (figure 3). By a smaller margin, still a majority of team practices and measures were considered to be facilitated efficiently in the respondents' respective organisations.

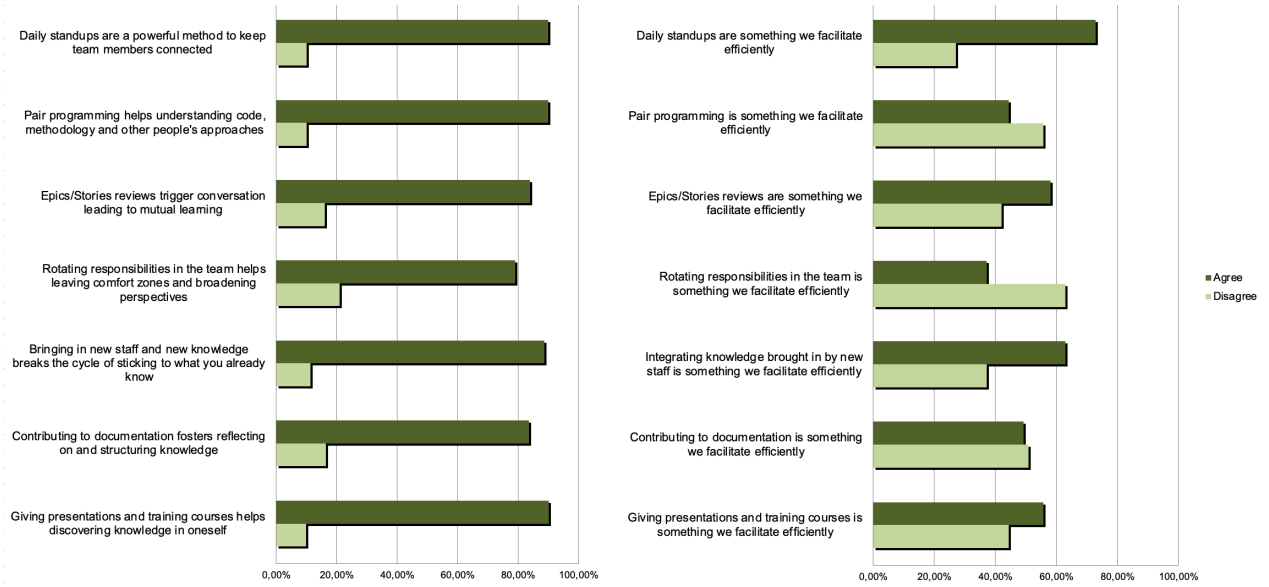

Figure 3. Assessment of some Team Practices and Measures

See external file: *teampractices.eps*

Recognised deficits in efficiency can be separated into two groups:

- A team practice or measure is considered not very important, like the above mentioned cross-disciplinary task forces and rotating of team responsibilities.

- A team practice or measure is considered important but not or seldomly used in one's organisation for some other reason.

Prominent examples for the second group include processes not directly contributing to a software project like participation in open source projects or cross-disciplinary exchange. Also, interestingly, pair programming, though found relevant by 90%, was considered to be facilitated efficiently by only 36.5%. However, while very useful from the knowledge management perspective, pair programming has not only advantages — it is exhausting for developers, stated 50% of the respondents.

Since most of the chosen team practices and measures originate from agile methodologies, the 'agile' vs. 'traditional' separation used in section 4.2 reveals interesting reasons. Employees from 'agile' organisations consider themselves more proficient in practices related to collaborative planning and knowledge transfer than their counterparts from 'traditional' organisations.

These findings answer the second part of RQ1 (how the chosen practices and measures used in software engineering organisations are assessed from the knowledge perspective).

4.4 How Team Practices and Measures correspond with Knowledge Activities

Practitioners were asked to 'tag' team practices and measures with knowledge activities executed or fostered by them. Table IV illustrates the results with respective bar lengths indicating relevance.

Part 1: Meetings	Acquire	Codify	Store	Maintain	Transfer	Create
Daily standups						
Recurring longer meetings						
Client Meetings						
Retrospectives						
Part 2: Development	**Acquire**	**Codify**	**Store**	**Maintain**	**Transfer**	**Create**
Design software						
Write Code						
Code Reviews						
Pair Programming						
Test each others' features						
Part 3: Planning	**Acquire**	**Codify**	**Store**	**Maintain**	**Transfer**	**Create**
Collaborative Release Planning						
Epics/Stories Reviews						
Joint estimate software features						
Part 4: Organisation	**Acquire**	**Codify**	**Store**	**Maintain**	**Transfer**	**Create**
Bring in new staff						
Onboarding / Mentoring						
Team assignment changes						
Rotate responsibilities in the team						
Cross-disciplinary task forces						
Part 5: Learning & Documentation	**Acquire**	**Codify**	**Store**	**Maintain**	**Transfer**	**Create**
Contribute to documentation						
Give presentations or training						
Engage in Special Interest Groups						
Attend Conferences						
Part 6: Other	**Acquire**	**Codify**	**Store**	**Maintain**	**Transfer**	**Create**
Informal exchange in the kitchen						
Informal gathering after work						
Part. in Open Source projects						
Part. in cross-disciplinary exchange						

Table IV. Team practices and Knowledge Activities

Generally, knowledge activities correspond with the 6 parts categorising practices and measures:

- The four kinds of meetings show relevance in maintaining and transferring knowledge while client meetings also help to acquire and create knowledge. Retrospectives even relate significantly to all knowledge activities with the only exception of acquiring knowledge.
- Development exhibits an emphasis on codifying, transferring and creating knowledge.
- Collaborative planning helps in transferring, also to a lesser degree maintaining and creating knowledge.
- Organisational measures leading to permanent or temporary changes in team composition emphasise knowledge transfer, acquisition and creation.
- Learning measures and documentation cover all 6 knowledge activities to different degrees.
- Informal exchange and participation in external activities are helpful for acquiring, transferring and creating knowledge.

Interestingly employees from 'agile' organisations were generally found to see more knowledge transfer and knowledge creation in all the chosen processes and activities.

The results, in particular the findings listed in table IV, answer the first part of RQ1 (which knowledge activities are related to which team practices and measures).

4.5 Most effective Team Practices and Measures

RQ2 asks how team practices and measures should be combined to achieve a mixture suitable for effective knowledge management in software engineering organisations. While table IV gives a good indication of how particular activities can be fostered, some team practices and measures are of particularly high value, either because of their relevance for some specific knowledge activities or their versatility:

- Pair programming is the most versatile practice, however one should consider that it can be exhausting for developers, see section 4.3.
- Cross-disciplinary task forces while being versatile offer both a lot of knowledge acquisition and transfer.
- Retrospectives received high scores for all knowledge activities but knowledge acquisition.
- Onboarding and mentoring is very powerful for both acquiring and transferring knowledge.
- Contributing to documentation is not only important for codifying and storing knowledge but also significantly contributes to maintaining and transferring knowledge.
- Rotating responsibilities in the team scores extremely high on knowledge transfer and also significantly in creating and maintaining knowledge while team assignment changes offering similar strengths at a slightly lower scale also bring in knowledge acquisition.

These few chosen examples lead to a number of general recommendations: Retrospectives should take place because of their versatility. While teams using Scrum are expected to hold them after every sprint, they are cancelled far too often due to time pressure as 63% confirmed. Pair programming can give a boost when needed but should not be overdosed. Teamwork is beneficial for knowledge, ideally when bringing in occasional changes, like rotating responsibilities or changing team assignments. Similarly, particular care when integrating new staff is highly profitable for both sides.

Of course just introducing some of these will hardly trigger an automatism not requiring any further steering or management: once a tool's impact is understood it should be applied in a way allowing it to fulfil its potential.

5 Discussion and Conclusion

This research aimed to make a step towards a holistic approach to knowledge management in software engineering by connecting some of its already existing elements to a well-established concept in knowledge management. This was achieved by using responses obtained from practitioners for linking the six knowledge activities proposed by Heavin and Adam (2012) to a set of development team practices and measures (most of which originating from agile methodologies) and assessing their respective relevance.

The results show that knowledge activities are not only important, but often already part of software teams' daily work, triggered and fostered by knowledge-related team practices and measures. A relation between the two worlds could be clearly established indicating its suitability as a tool for decision making by project leaders in software engineering. This is of significant value to practitioners as their traditional approach is usually detached from classic knowledge management research. Practitioners' feedback obtained by the corresponding author after presentations or workshops on knowledge management in software engineering have frequently confirmed these findings' relevance and importance.

While most of the chosen practices and measures originate from the world of agile methodologies, the responses indicate that this does not restrict their applicability: practically all their respective organisations rely on agile frameworks like Scrum or the hybrid approach to some degree. Hence, chosen practices and measures can be used for supporting knowledge management without sacrificing productivity, as developers are already familiar with them (also the already-established approach of inspection and adaptation can be used for fine-tuning their KM impact). On a side note, this finding also confirms other researchers' findings of agile software development's particular suitability for knowledge management because of their focus on communication and collaboration (Dissanayake *et al.* 2013; Heavin and Adam, 2014; C. Khalil and S. Khalil, 2019).

While the practitioners' responses recognised the knowledge activities' relevance they assessed their respective organisations' efficiency in facilitating them rather critically. Understanding how elements of the teams' daily work relate to them can be a key to solving this problem. Some of the practices' and measures' versatility in terms of knowledge has another positive effect: team leaders do not need to rely on a large number of them for achieving the desired mixture of knowledge activities.

Software project work is frequently subject to enormous time pressure. Teams dealing with knowledge efficiently have an advantage here, while unfortunately systematic knowledge management usually does not promise immediate return on investment. This is why blending well with existing work and not coming at an additional cost are critical requirements. Relying on commonly-used procedures with some direct value to the software development process while understanding their potential in knowledge management fills this gap.

These outcomes may already sound familiar to experienced practitioners, however, increased attention, a more substantiated understanding of practices and measures and their connection to knowledge-related aspects can contribute significantly to more sustainable team development with little or no overhead.

Future research should complement this work's findings by a quantitative study allowing to confirm the results' representativeness. Also the connections' significance displayed in Table XXIX suggest the construction of scores in a framework for calculating key performance indicators (KPIs) on teams' existing application of knowledge activities. How such a framework will be constructed as well as its applicability in implementing knowledge management in software teams is both of academic interest and practical relevance.

References

Abdur, R., T. Bhuiyan and R. Ahmed, (2015), "Knowledge management in distributed agile software development projects", *Proceedings of the AIKM 2014 - Artificial Intelligence for Knowledge Management*, vol. 469, pp. 107–131.

Abrahamsson, P. *et al.* (2002), "Agile software development methods: review and analysis", *Proc. Espoo 2002*, pp. 3–107.

Agile Alliance *et al.* (n.d.), What is hybrid agile, anyway?, `https://www.agilealliance.org/what-is-hybrid-agile-anyway/`, Accessed on 13.01.2021.

Alecu, F., (2011), "Managing software development projects, the sequence of the project phases", *Oeconomics of Knowledge* 3 (4), pp. 24–30.

Andersen, A., (1996), The knowledge management tool: external benchmarking version, American Productivity and Quality Center, Houston.

Andreeva, T. and I. Ikhilchik, (2011), "Applicability of the SECI Model of knowledge creation in Russian cultural context: theoretical analysis", *Scholarly Articles at Russian Management Journal* 18 (1), DOI: `10.1002/kpm.351`.

Aurum, A., F Daneshgar and J. Ward, (2008), "Investigating knowledge management practices in software development organisations - an Australian experience", *Information and Software Technology* 50 (6), pp. 511–533, DOI: `10.1016/j.infsof.2007.05.005`.

Basri, S. and R. O'Connor, (2011), "The impact of software development team dynamics on the knowledge management process(S).", *Proceedings of the 23rd International Conference on Software Engineering and Knowledge Engineering*, pp. 339–342.

Beck, K., (1999), "Embracing change with extreme programming", *Computer* 32 (10), pp. 70–77, DOI: `10.1109/2.796139`.

Beck, K. *et al.* (2001), Manifesto for agile software development, available at: `http://www.agilemanifesto.org/` (accessed 6 February 2021).

Behutiye, W. *et al.* (2019), "Management of quality requirements in agile and rapid software development: a systematic mapping study", *Information and Software Technology* 123, DOI: `10.1016/j.infsof.2019.106225`.

Benington, H. D., (1983), "Production of large computer programs", *Annals of the History of Computing* 5 (4), pp. 350–361.

Bierly, P., E. Kessler and E. Christensen, (2000), "Organizational learning, knowledge and wisdom", *Journal of Organizational Change Management* 13 (6), pp. 595–618.

Bjørnson, F. O. and T. Dingsøyr, (2008), "Knowledge management in software engineering: a systematic review of studied concepts, findings and research methods used", *Information and Software Technology* 50 (11), pp. 1055–1068.

Bryde, D. *et al.* (2018), "KM and project management", Syed, J. *et al. The Palgrave Handbook of Knowledge Management*, pp. 539–561.

Bryman, A. and E. Bell, (2007), Business research methods, 2nd edition, Oxford University Press, Oxford.

Collins, Harry, (2010), "Tacit & explicit knowledge", *Bibliovault OAI Repository, the University of Chicago Press*, DOI: `10.7208/chicago/9780226113821.001.0001`.

Costa, Vítor and Samuel Monteiro, (2017), "Key knowledge management processes for innovation: a systematic literature review", *VINE Journal of Information and Knowledge Management Systems* 46 (3), pp. 386–410, DOI: `10.1108/VJIKMS-02-2015-0017`.

Creswell, J.W., (2014), Research design: qualitative, quantitative, and mixed methods approaches, 4th edition, Sage Publications, Los Angeles.

Dietze, M. and M. Kahrens, (2021), "Transform knowledge assets into reality: How the purposeful combination of knowledge activities enables organisations to channelise the knowledge flow in software engineering development", *Proceedings of the IFKAD, 1-3 September 2021, Rome, Italy.*

Dima, A. and M. A. Maassen, (2018), "From waterfall to agile software: development models in the IT sector, 2006 to 2018. Impacts on company management", *Journal of International Studies* 11 (02), pp. 315–326, DOI: `10.14254/2071-8330.2018/11-2/21`.

Dingsøyr, T. and R. Conradi, (2002), "A survey of case studies of the use of knowledge management in software engineering", *International Journal of Software Engineering and Knowledge Engineering* 12 (4), pp. 391–414, DOI: 10.1142/S0218194002000962.

Dissanayake, I., Ramakrishna Dantu and S. Nerur, (2013), "Knowledge management in software development: the case of agile software", *Proceedings of the 19th Americas conference on information systems, AMCIS 2013 - hyperconnected world: anything, anywhere, anytime*, vol. 3, pp. 2298–2304.

Earl, M., (2001), "Knowledge management strategies: toward a taxonomy", *Journal of Management Information Systems* 18 (1), pp. 215–242.

Eisenhardt, K.M. and M.E. Graebner, (2007), "Theory building from cases: opportunities and challenges", *The Academy of Management Journal* 50 (1), pp. 25–32, DOI: 10.5465/AMJ.2007.24160888.

Grover, V. and T. H. Davenport, (2017), "General perspectives on knowledge management: fostering a research agenda", *Journal of Management Information Systems* 18 (1), pp. 5–21.

Heavin, C. and F. Adam, (2012), "Characterising the knowledge approach of a firm: an investigation of knowledge activities in five software SMEs", *Electronic Journal of Knowledge Management* 10 (11), pp. 48–63.

Heavin, C. and F. Adam, (2014), "From knowledge activities to knowledge scenarios: cases in five irish software SMEs", *International Journal of Management and Enterprise Development* 13 (1), pp. 37–61, DOI: 10.1504/IJMED.2014.059852.

Hegde, R. and G. Walia, (2014), "How to enhance the creativity of software developers: a systematic literature review", *Proceedings of the 26th International Conference on Software Engineering and Knowledge Engineering*.

Holsapple, C. and K. D. Joshi, (2004), "A knowledge management ontology", Holsapple, C., *Handbook on Knowledge Management 1: knowledge Matters*, Springer, Berlin, Heidelberg, pp. 89–124.

Holste, J. S. and D. Fields, (2010), "Trust and tacit knowledge sharing and use", *Journal of Knowledge Management* 14 (1), pp. 128–140.

Ichijo, K. and I. Nonaka, (2008), Knowledge creation and management: new challenges for managers, Oxford University Press, New York.

Imani, T., (2017), "Does a hybrid approach of agile and plan-driven methods work better for IT system development projects?", *International Journal of Engineering Research and Applications* 07 (03), pp. 39–46, DOI: 10.9790/9622-0703043946.

Jeon, S. *et al.* (2011), "Quality attribute driven agile development", *Proceedings of the Ninth International Conference on Software Engineering Research, Management and Applications*, pp. 203–210.

Jetter, A. *et al.* (2006), Knowledge integration: the practice of knowledge management in small and medium enterprises, Physica, Heidelberg, pp. 1–203.

Juric, R., (2000), "Extreme programming and its development practices", *Proceedings of the 22nd International Conference on Information Technology*, pp. 97–104.

Kahrens, M. and D. H. Früauff, (2018), "Critical evaluation of Nonaka's SECI model", Syed, J. *et al. The Palgrave Handbook of Knowledge Management*, Springer International Publishing, Cham, pp. 53–83.

Kavitha, R. and M. S. Irfan, (2011), "A knowledge management framework for agile software development teams", DOI: 10.1109/PACC.2011.5978877.

Khalil, C. and S. Khalil, (2019), "Exploring knowledge management in agile software development organizations", *International Entrepreneurship and Management Journal*, DOI: 10.1007/s11365-019-00582-9.

Kraaijenbrink, J., D. Faran and A. Hauptman, (2006), "Knowledge integration by SMEs — framework", *Knowledge Integration*, Springer, Berlin, Heidelberg, chap. 2, pp. 17–28, DOI: 10.1007/3-7908-1681-7_2.

Laplante, P.A., (2007), What every engineer should know about software engineering, What Every Engineer Should Know, Taylor & Francis, Boca Raton.

Lee, C. S. and R. S. Kelkar, (2013), "ICT and knowledge management: perspectives from the SECI model", *The Electronic Library* 31 (2), pp. 226–243.

Lin, D., (2021), "Wissensmanagement Reloaded - ein Ordnungsrahmen für den systemischen Umgang mit Wissen im Enterprise 2.0", ger, MA Thesis, Technische Universität Dresden, 2010, MA thesis, Kiel, Hamburg, available at: http://hdl.handle.net/10419/36700 (accessed 6 February 2021).

Little, T., (2005), "Context-adaptive agility: managing complexity and uncertainty", *Software, IEEE* 22 (3), pp. 28–35, DOI: 10.1109/MS.2005.60.

Maiti, R. and F. Mitropoulos, (2015), "Capturing, eliciting, predicting and prioritizing (CEPP) non-functional requirements metadata during the early stages of agile software development", *Proceedings of the IEEE SoutheastCon*, vol. 2015, DOI: 10.1109/SECON.2015.7133007.

Mansoor, M. *et al.* (2019), "Adaptation of modern agile practices in global software engineering", Rehman, M. *et al. Human Factors in Global Software Engineering*, IGI Global.

Modern agile web site (n.d.), https://modernagile.org, Accessed on 13.01.2021.

Nezafati, N., A. Afrazeh and S. M. J. Jalali, (2009), "A dynamic model for measuring knowledge level of organizations based on Nonaka and Takeuchi model (SECI)", *Scientific Research and Essays*, pp. 531–542.

Nonaka, I. and N. Konno, (1998), "The concept of "ba": building a foundation for knowledge creation", *California Management Review* 40 (3), pp. 40–54.

Nonaka, I. and H. Takeuchi, (1995), The knowledge-creating company: how Japanese companies create the dynamics of innovation, Oxford University Press, New York.

Nonaka, I. and R. Toyama, (2003), "The knowledge-creating theory revisited: knowledge creation as a synthesizing process", *Knowledge Management Research & Practice* 1 (1), pp. 2–10.

Nonaka, I. and G. von Krogh, (2009), "Tacit knowledge and knowledge conversion: controversy and advancement in organizational knowledge creation theory", *Organization Science* 20 (3), pp. 635–652.

Palmer, S. R. and M. Felsing, (2001), A practical guide to feature-driven development, 1st edition, Pearson Education, Upper Saddle River.

Pentland, B. T., (1995), "Information systems and organizational learning: the social epistemology of organizational knowledge systems", *Accounting, Management and Information Technologies* 5 (1), Special Issue Information Technology and Organizational Learning, pp. 1–21, ISSN: 0959-8022, DOI: 10.1016/0959-8022(95)90011-X.

Project Management Institute, (2013), A guide to the project management body of knowledge: PMBOK guide, PMBOK® Guide Series, Project Management Institut Inc, Newtown Square.

Raval, R. and H. Rathod, (2014), "Improvements in agile model using hybrid theory for software development in software engineering", *International Journal of Computer Applications* 90 (16), pp. 26–31, DOI: 10.5120/15806-4677.

Rodríguez, P. *et al.* (2012), "Survey on agile and lean usage in Finnish software industry", *Proceedings of the International Symposium on Empirical Software Engineering and Measurement*, pp. 139–148.

Rus, I. and M. Lindvall, (2002), "Knowledge management in software engineering", *IEEE Software* 19 (3), pp. 26–38.

Saifi, S., S. Dillon and R. Mcqueen, (2016), "The relationship between management support and knowledge sharing: an exploratory study of manufacturing firms: management support and knowledge sharing", *Knowledge and Process Management* 23 (2), pp. 124–135, DOI: 10.1002/kpm.1506.

Sandhawalia, B. and D. Dalcher, (2010), "Knowledge flows in software projects: an empirical investigation", *Knowledge and Process Management* 17 (4), pp. 205–220, DOI: 10.1002/kpm.357.

Sapir, A., I. Drori and S. Ellis, (2016), "The practices of knowledge creation: collaboration between peripheral and core occupational communities", *European Management Review* 13 (1), pp. 19–36, DOI: 10.1111/emre.12064.

Scharmer, C. O., (2001), "Self-transcending knowledge: sensing and organizing around emerging opportunities", *Journal of Knowledge Management* 5 (2), pp. 137–151.

Schreyögg, G. and D. Geiger, (2003), "Kann die Wissensspirale Grundlage des Wissensmanagements sein?", *Diskussionsbeiträge des Instituts für Management, Nr. 20, Berlin.*

Schreyögg, G. and D. Geiger, (2004), "Kann man implizites in explizites Wissen konvertieren? Die Wissensspirale auf dem Prüfstand", Frank, U., *Wissenschaftstheorie in Ökonomie und Wirtschaftsinformatik*, Springer, Heidelberg, pp. 269–288.

Schwaber, K. and M. Beedle, (2001), "Agile software development with scrum".

Shongwe, M. M., (2017), "Knowledge management in small software development organisations: a South African perspective", *SA Journal of Information Management* 19 (1).

Silverman, D., (2014), Interpreting qualitative data, 5th edition, Sage Publications.

Sommerville, I., (2008), Software engineering, 8th edition, Pearson, Harlow.

Stenmark, D., (2000), "Leveraging tacit organizational knowledge", *Journal of Management Information Systems* 17 (3), pp. 9–24.

Sungkur, R. and M. Ramasawmy, (2014), "Knowledge4Scrum, a novel knowledge management tool for agile distributed teams", *VINE* 44 (3), pp. 394–419, DOI: 10.1108/VINE-12-2013-0068.

Sutherland, J. and K. Schwaber, (2010), Scrum guide, available at: https://scrumguides.org/scrum-guide.html (accessed 6 February 2021).

Takeuchi, H. and I. Nonaka, (1986), "The new new product development game", *Harvard Business Review*.

Tee, M. Y. and S. S. Lee, (2013), "Advancing understanding using Nonaka's model of knowledge creation and problem-based learning", *International Journal of Computer-Supported Collaborative Learning* 8 (3), pp. 313–331.

Teece, D., (2010), "Technological innovation and the theory of the firm: the role of enterprise-level knowledge, complementarities, and (dynamic) capabilities", *Handbook of the Economics of Innovation* 1, pp. 679–730.

Umbreen, M., J. Bukhari and S. Shaheed, (2015), "A comparative approach for SCRUM and FDD in agile", *International Journal of Computer Science and Innovation* 2015 (02), pp. 79–87.

van Manen, M., (2007), "Phenomenology of practice", *Phenomenology & Practice* 1 (1), pp. 11–30, DOI: 10.29173/pandpr19803.

van Manen, M., (2016), Researching lived experience: human science for an action sensitive pedagogy, 2nd edition, Routledge, New York.

Vasanthapriyan, S., J. Tian and J. Xiang, (2015), "A survey on knowledge management in software engineering", *Proceedings of the 2015 IEEE International Conference on Software Quality, Reliability and Security - Companion*, pp. 237–244.

Viana, D. *et al.* (2013), "A qualitative study about the life cycle of lessons learned", *Proceedings of the 2013 6th International Workshop on Cooperative and Human Aspects of Software Engineering, CHASE 2013*, pp. 73–76.

von Krogh, G., K. Ichijo and I. Nonaka, (2000), Enabling knowledge creation: how to unlock the mystery of tacit knowledge and release the power of innovation, Oxford University Press, New York.

von Krogh, G., I. Nonaka and L. Rechsteiner, (2012), "Leadership in organizational knowledge creation: a review and framework", *Journal of Management Studies* 49 (1), pp. 240–277.

Ward, J. and A. Aurum, (2004), "Knowledge management in software engineering - describing the process", *Proceedings of the 2004 Australian Software Engineering Conference*, pp. 137–146.

6 Appendix: The Questionaire

6.1 Demographic Questions

	Team Lead	Developer	Architect	Dev Ops	Other
What best describes your role in your organisation?					

	Projects	Products / Services	Both
Which of these two does your company develop?			

	1..5	6..10	11..15	16..20	> 20
What is the typical size of a software development team in your company?					

	0%	25%	50%	75%	100%
How many developers in your organisation are usually assigned to more than one project at a time?					

	1..10	11..50	51..250	250..500	> 500
How many employees has your company?					

6.2 Development Methodologies

How much are the following methods used in your company?

	0%	25%	50%	75%	100%
Scrum					
Other agile					
Waterfall					
Hybrid					
Others					

6.3 Knowledge Activities

Organisational knowledge creation requires systematic effort.

	Strongly disagree	Disagree	Tend to disagree	Tend to agree	Agree	Fully agree
This is something I consider extremely important for our work						
This is something that my company addresses well						

In our work we always need to gather new knowledge

	Strongly disagree	Disagree	Tend to disagree	Tend to agree	Agree	Fully agree

This is something I consider extremely important for our work						
This is something that my company addresses well						

We need to put effort in making our knowledge comprehensible

	Strongly disagree	Disagree	Tend to disagree	Tend to agree	Agree	Fully agree
This is something I consider extremely important for our work						
This is something that my company addresses well						

Extracted knowledge needs to be persisted so that others can access it

	Strongly disagree	Disagree	Tend to disagree	Tend to agree	Agree	Fully agree
This is something I consider extremely important for our work						
This is something that my company addresses well						

Knowledge requires effort to be updated on a regular basis

	Strongly disagree	Disagree	Tend to disagree	Tend to agree	Agree	Fully agree
This is something I consider extremely important for our work						
This is something that my company addresses well						

Passing on knowledge requires a strategy and well-designed methods

	Strongly disagree	Disagree	Tend to disagree	Tend to agree	Agree	Fully agree
This is something I consider extremely important for our work						
This is something that my company addresses well						

New knowledge can evolve as a result of all this

	Strongly disagree	Disagree	Tend to disagree	Tend to agree	Agree	Fully agree

This is something I consider extremely important for our work						
This is something that my company addresses well						

6.4 Team Practices and Measures

Meetings

	Strongly disagree	Disagree	Tend to disagree	Tend to agree	Agree	Fully agree
Daily standups are a powerful method to keep team members connected						
Daily standups are something we facilitate efficiently						
Recurring longer meetings are a space for open and inspiring debate						
Recurring longer meetings are something we facilitate efficiently						
Client meetings force team members to view problems from a different angle						
Client meetings are something we facilitate efficiently						
Retrospectives help integrating new approaches to solving problems						
Retrospectives tend to get cancelled due to time and project pressure						
Retrospectives are something we facilitate efficiently						

Development

	Strongly disagree	Disagree	Tend to disagree	Tend to agree	Agree	Fully agree
Code reviews lead to productive conversation						
Code reviews are something we facilitate efficiently						
Pair programming helps understanding code, methodology and other people's approaches						
Pair programming is exhausting for developers						
Pair programming is something we facilitate efficiently						

Testing each others' features fosters mutual learning						
Testing each others' features is something we facilitate efficiently						

Planning

	Strongly disagree	Disagree	Tend to disagree	Tend to agree	Agree	Fully agree
Collaborative release planning fosters understanding the project beyond one's own role						
Collaborative release planning is something we facilitate efficiently						
Epics/Stories reviews trigger conversation leading to mutual learning						
Epics/Stories reviews are something we facilitate efficiently						
Joint estimating software features fosters discussion on what really needs to be built and risks						
Joint estimating software features is something we facilitate efficiently						

Organisation

	Strongly disagree	Disagree	Tend to disagree	Tend to agree	Agree	Fully agree
Bringing in new staff and new knowledge breaks the cycle of sticking to what you already know						
Integrating knowledge brought in by new staff is something we facilitate efficiently						
Onboarding / mentoring fosters mutual learning in a stress-free environment						
Onboarding / mentoring is something we facilitate efficiently						

Team assignment changes help fighting stalled learning and sectarianism in long-existing teams						
Team assignment changes are something we facilitate efficiently						
Rotating responsibilities in the team helps leaving comfort zones and broadening perspectives						
Rotating responsibilities in the team is something we facilitate efficiently						
Cross-disciplinary task forces foster innovation by combining and developing new approaches						
Cross-disciplinary task forces are something we facilitate efficiently						

Learning & Documentation

	Strongly disagree	Disagree	Tend to disagree	Tend to agree	Agree	Fully agree
Contributing to documentation fosters reflecting on and structuring knowledge						
Contributing to documentation is something we facilitate efficiently						
Giving presentations and training courses helps discovering knowledge in oneself						
Giving presentations and training courses is something we facilitate efficiently						
Engagement in Special Interest Groups fosters mutual learning and expertise						
Engagement in Special Interest Groups is something we do						
Attending conferences yields inspiration and new knowledge						
Attending conferences is something we do						

Other

	Strongly disagree	Disagree	Tend to disagree	Tend to agree	Agree	Fully agree
Informal exchange in the kitchen is a company's most powerful knowledge reactor						
Informal exchange in the kitchen is something we do a lot						
Informal gathering after work fosters cross-disciplinary thinking						
Informal gathering after work is something we do						
Participation in Open Source projects brings new skills and knowledge into the team						
Participation in Open Source projects is something we facilitate efficiently						
Participation in cross-disciplinary exchange helps buillding networks of real value						
Participation in cross-disciplinary exchange is something we facilitate efficiently						

6.5 Linking Knowledge Activities to Team Practices and Measures

Meetings

	Acquire	Codify	Store	Maintain	Transfer	Create
Daily standups						
Recurring longer meetings						
Client meetings						
Retrospectives						

Development

	Acquire	Codify	Store	Maintain	Transfer	Create
Design software						
Write Code						
Code Reviews						
Pair Programming						
Test each others' features						

Planning

	Acquire	Codify	Store	Maintain	Transfer	Create
Collaborative Release Planning						

Epics/Stories Reviews						
Joint estimate software features						

Organisation

	Acquire	Codify	Store	Maintain	Transfer	Create
Bring in new staff and new knowledge						
Onboarding / Mentoring						
Team assignment changes						
Rotate responsibilities in the team						
Cross-disciplinary task forces						

Learning & Documentation

	Acquire	Codify	Store	Maintain	Transfer	Create
Contribute to documentation						
Give presentations or training courses						
Engage in Special Interest Groups						
Attend Conferences						

Other

	Acquire	Codify	Store	Maintain	Transfer	Create
Informal exchange in the kitchen						
Informal gathering after work						
Participation in Open Source projects						
Participation in cross-disciplinary exchange						