Understanding Knowledge Management in Agile Software Development Practice

Yanti Andriyani[1(✉)], Rashina Hoda[1], Robert Amor[2]

[1]SEPTA Research, Department of Electrical and Computer Engineering,
The University of Auckland, Auckland, New Zealand
yand610@aucklanduni.ac.nz
r.hoda@auckland.ac.nz

[2]Department of Computer Science, The University of Auckland, Auckland, New Zealand
trebor@cs.auckland.ac.nz

Abstract: Knowledge management in agile software development has typically been treated as a broad topic resulting in major classifications of its schools and concepts. What inherent knowledge is involved in everyday agile practice and how agile teams manage it is not well understood. To address these questions, we performed a Systematic Literature Review of 48 relevant empirical studies selected from reputed databases. Using a thematic analysis approach to the synthesis, we discovered that (a) agile teams use three knowledge management strategies: discussions, artifacts and visualisations to manage knowledge (b) there are three types of software engineering knowledge: team progress as project knowledge; requirements as product knowledge; and coding techniques as process knowledge. (c) this knowledge is presented in several everyday agile practices. A theoretical model describing how knowledge management strategies and knowledge types are related to agile practices is also presented. These results will help agile practitioners become aware of the specific knowledge types and knowledge management strategies and enable them to better manage them in everyday agile practices. Researchers can further investigate and build upon these findings through empirical studies.

Keywords: Agile Software Development Practice, Knowledge Type, Knowledge Management Strategies

1 Introduction

Agile Software Development (ASD) methods such as Scrum and Extreme Programming (XP) ushered in an era of lightweight software development [1]. Unlike traditional software methods such as waterfall, which was driven by detailed specifications and

[1]Corresponding Author: Building 903, 386 Khyber Pass, New Market Auckland 1023, New Zealand. Tel: +64 9 923 1377 Email address: yand610@aucklanduni.ac.nz (Y. Andriyani), r.hoda@auckland.ac.nz (R. Hoda), trebor@cs.auckland.ac.nz (R. Amor)

design upfront and involved rigorous documentation [2] as a process of managing knowledge, agile methods emphasize social interactions and collaboration among team members in applying and sharing knowledge.

Prior reviews and investigations of knowledge management in ASD have typically treated it as a broad topic resulting in major classifications of its schools, concepts [3], strengths, weaknesses, opportunities and threats [4]. However, what inherent knowledge is involved in everyday agile practice and how agile teams manage this knowledge is not well understood. To address this gap, we performed a Systematic Literature Review (SLR) involving 48 relevant empirical studies [5] filtered from an initial pool of 2317 articles selected from the reputed academic databases of Springer, Scopus, and IEEE Xplore.

This SLR aims to provide an overview of the studies on this particular topic by answering the following research questions (RQ):

RQ1: Which specific agile practices support knowledge management?

RQ2: What is the inherent knowledge involved in these agile practices and how do agile teams manage that knowledge?

The next section provides an overview of background and related work of this research. Section 3 gives the research method used and section 4 discusses the results of this study. Section 5 provides the discussions of the findings and section 6 provides the conclusion.

2 Background and Related Work

The relevant ASD and knowledge management concept definitions and summaries of prior reviews are presented in this section to aid in the understanding of the findings described later.

2.1 Knowledge Classifications

Knowledge is defined as: *"A fluid mix of framed experience, values, contextual information, and expert insight that provides a framework for evaluating and incorporating new experiences and information"* [6]. Knowledge is described in two basic forms: tacit knowledge or the knowledge that is implicit and not clarified in an accessible form; and explicit knowledge or the knowledge that is visualised or clarified in accessible forms such as writing on sticky notes, drawing pictures to describe the processes or feelings, drawing progress charts, etc. [7].

A classification of knowledge in software engineering describes three types of knowledge: project, product and process knowledge [8]. Project knowledge is defined as *"the knowledge about resources, functional and attributes requirements, work products, budget, timing, milestones, deliverables, increments, quality targets and performance parameters"* [8]. Documentation and resources include contracts and project plans based on requirement designs; and the parameters are analysed through comparisons between the planned and actual cost, effort and time [9].

Product knowledge is defined as *"the knowledge about [the] product features and how they relate to other products, standards, protocols and the like"* [8]. Specifically, product knowledge is related to the product features, its interface and its dependency on technology (e.g. specific programming language or platform), network configurations, standards (related to components of the product) and protocols.

Process knowledge is defined as *"the knowledge about business processes, workflows, responsibilities, supporting technologies and interfaces between processes"* [8]. In other words, process knowledge comprises of work targets and task information retrieved from the business model; the workflow and responsibilities referring to how the various artifacts are produced and who is responsible for specific tasks and how they accomplish them based on the milestones [8].

2.2 Prior Reviews on Knowledge Management in ASD

Knowledge management in organizations is defined as: *"A method that simplifies the process of sharing, distributing, creating, capturing and understanding the company knowledge"* [6]. Knowledge management in traditional software development involved the use of various documents to capture and represent the knowledge related to the various stages of software development life cycles [3]. In contrast to traditional methods, agile methods emphasize tacit knowledge over explicit knowledge, relying on individual, team and customer communications and interactions [4].

Prior reviews and investigations on knowledge management in software engineering in general and ASD, in particular, have classified knowledge schools [10] and concepts [3]. Bjørnson & Dingsøyr [10] classified two types of knowledge management schools in software engineering. The first category is the *technocratic* school, which refers to knowledge management strategies that focus on explicating knowledge and its flows. The second category is the *behavioural* school, which focuses on collaboration and communication as knowledge management strategies.

Yanzer et al. [3] presented several concept maps about knowledge management in agile projects. The first map covers ways of communication that focus on techniques and tools to manage the conversation. The second is about human and social factors, which discusses knowledge management adoption in agile projects and knowledge artifact usage in the projects. Other concepts include tools for knowledge management and knowledge representation forms related to managing tacit knowledge, and emphasize the managing of tacit knowledge by using tools to clarify the knowledge.

Analysing the prior reviews on knowledge management in ASD, we could see that there was a need for a specific explanation of knowledge management in ASD from the viewpoint of daily agile practice.

3 Review Method

A Systematic Literature Review (SLR) aims to collect evidence from the prior literature based on research questions to provide guidelines for practitioners [11]. We followed the specific steps of performing a SLR as recommended by Kitchenham [11], such as planning, study selection, data extraction and synthesis, and reporting the review.

3.1 Planning the Review and Identifying Relevant Literature

We constructed a search strategy to identify relevant literature by deriving major terms from the research questions, listing the keywords and developing search strings from these major terms and their synonyms using AND/OR operators. The search string that we used was: *("knowledge manag*" OR "learning manag*") AND ("agile" OR "scrum" OR "XP" OR "Lean") AND ("software" AND "team")*

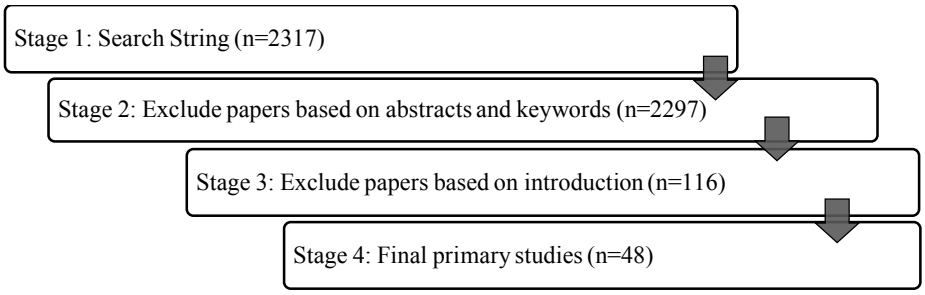

Fig. 1. SLR Screening Process

The search string was used to filter articles from three reputed academic databases:

- Springer, http://www.springer.com
- Scopus, www.scopus.com
- IEEE Xplore, http://www.ieeexplore.ieee.org

A total of 2317 papers were obtained initially, of which 195 papers were from Scopus based on title and abstracts while Springer Link resulted in 2097 papers and 25 results were from IEEE Xplore.

3.2 Publication Selection

Inclusion and exclusion criteria (described in [5]) were designed to help select from the initial pool of papers resulting from the searches. The results from the search string generated 2317 papers, which were screened in the next stage (see Fig. 1). The second stage generated 2297 papers, which screened the papers based on how the abstract and keywords related to the RQs and the inclusion and exclusion criteria [5]. Stage 3 involved reading the introduction section of the 2297 papers checked against the inclusion and exclusion criteria and resulted in 116 papers being filtered. All 116 papers were read in stage 4 resulting in a total of 48 papers selected as the primary studies based on the inclusion and exclusion criteria. Most papers were excluded because they only provided theoretical explanations and no empirical evidence.

3.3 Data Extraction and Synthesis

Data extraction involved extracting detailed information from the 48 primary studies, such as the paper citation details, answers to the review questions and the main stufy findings into an excel sheet. The emphasis was placed on extracting evidence to support the review questions. The first author was responsible for data extraction overseen by the co-authors who provided guidance throughout the process and helped reach consensus in certain cases.

Data was analysed by determining themes from the selected papers using thematic analysis [12]. Initial codes summarizing key ideas were selected after reading the primary studies thoroughly. For example, the specific artifacts used in ASD, such as product backlog, user stories etc. were collectively classified as *artifacts* since they contained useful knowledge about the software requirements. Similarly, UML modeling, burn down charts etc. were classified as *visualisations*.

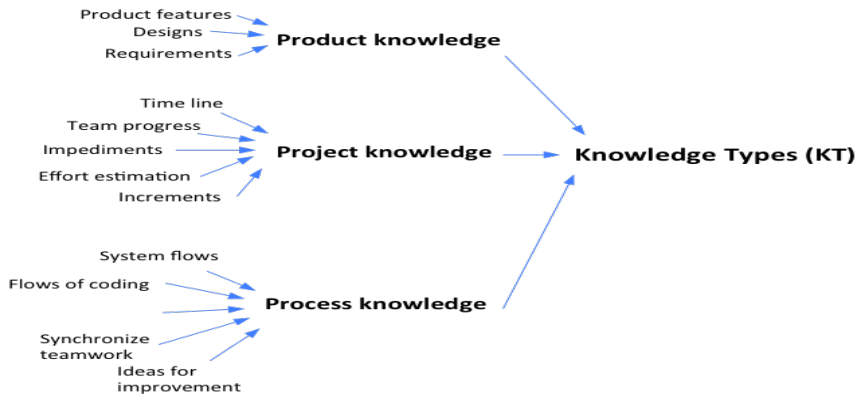

Fig. 2. Emergence of Knowledge Types (KT) Theme Categories

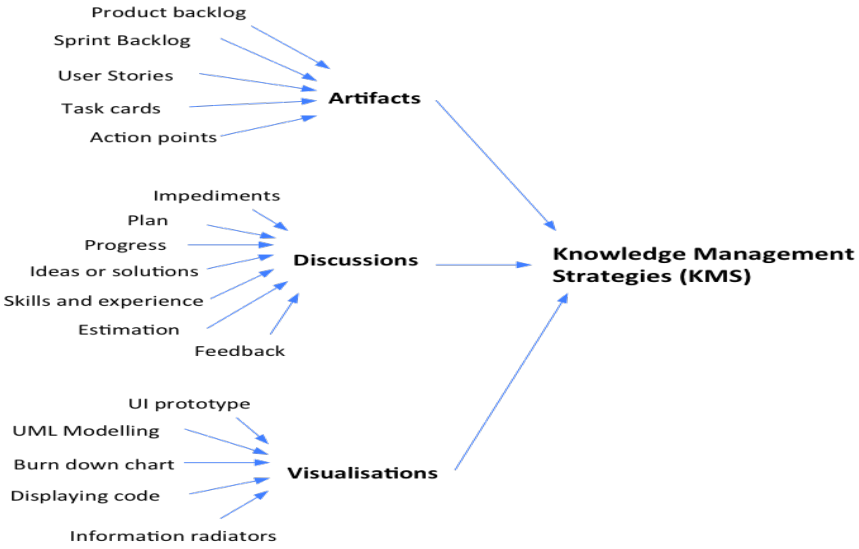

Fig. 3. Emergence of Knowledge Management Strategies (KMS) Theme Categories

Themes were derived as the next level of abstraction, gathering similar codes together [12]. For example, the codes artifacts, visualisation and discussion collectively formed the theme knowledge management strategies since each of them is a type of knowledge management strategy. Fig. 2 and 3 depict the emergence of the themes *'knowledge management strategies' (KMS)* and *'knowledge types' (KT)* from the underlying codes. Another theme that emerged naturally from the codes was *'agile practices' (AP)* which included the specific agile practices reported in the primary studies.

4 Results

4.1 Agile Practices Supporting Knowledge Management

In response to *RQ1 "Which specific agile practices support knowledge management?"* we identified the following agile practices to support knowledge management [5]:

- **Scrum Practices**: Sprint/Release Planning [S11,S13,S14,S20,S24,S38-S40], Daily Scrum [S6,S7,S10-S12,S23,S24,S41-S43], and Sprint Retrospective [S11,S13,S16,S24,S44-S46].
- **XP Practices:** Pair Programming [S3-S5,S7,S17,S19,S22,S48], Refactoring [S15,S17,S19,S25], and Planning Game [S16,S17,S19,S27].

The agile practices above support knowledge management practices through activities such as discussions, artifacts and visualisations [5]. The most commonly referred to agile practice was daily scrum across ten primary studies, followed by sprint/release planning meeting and pair programming across eight primary studies each and sprint retrospective across seven primary studies. Refactoring and the planning game were referred to by four studies each.

Some agile practices were not covered, such as sprint review, product backlog refinement, testing and small releases. One possible explanation is that the product backlog refinement is a relatively new, optional and lesser known Scrum practice [13]. The knowledge involved in other practices such as the sprint review, testing, and small releases, focuses more on the product knowledge which is presented as a deliverable product or a product increment in a 'demo' meeting [13], and released in small units of functionality to customers [1].

Although the sprint review, product backlog refinement, small releases, and testing were not covered in the primary studies, it does not imply that they do not involve any knowledge management. Knowledge based on product refinement, re-prioritisation, customer feedback, team reviews, are all valuable knowledge that needs to be managed by agile teams. Thus, further investigation is required to understand how agile teams use, manage and refer to the knowledge involved in these practices.

4.2 Knowledge Involved in Agile Practices

In response to RQ2: *"What is the inherent knowledge involved in these agile practices and how do agile teams manage that knowledge?"* we discovered that the three types of software engineering knowledge – product, project, and process knowledge [8] were captured in everyday agile practices.

Table 1 presents a summarized view of the knowledge types involved in agile practices. These knowledge types are managed in six agile practices. For example, product knowledge found to be involved in the release and sprint planning meetings included domain context [S6] and product features (systems requirements) [S12], which facilitates agile teams to gain knowledge about the product to be developed. In daily standup, a wall or Scrum board [S12] is used to stick up story cards that contain several tasks, which are broken-down from the product backlog.

Table 1. Knowledge Types (KT) in ASD (based on [8])

Knowledge Types (KT) based on [8]	***Description [8]***	***Examples in ASD practices***
Product Knowledge	*"The knowledge about [the] product features and how they relate to other products, standards, protocols and the like*.	Domain context [S6]; product features on user stories [S12]; coding [S12], testing [S11].
Project knowledge	*"The knowledge about resources, functional and attributes requirements, work products, budget, timing, milestones, deliverables, increments, quality targets and performance parameters"*	Project/daily goals [S6]; timeline [S10]; progress line; lack of time for testing and work targets [S10]
Process Knowledge	*"The knowledge about business processes, workflows, responsibilities, supporting technologies and interfaces between processes"*	Systems flows [S11]; business process [S11]; other team member's role and their interdependencies [S6]; synchronizing teamwork; ideas of improvement [S24]; workflow of coding and working code [S19].

Project knowledge is managed in several agile practices, such as sprint/release planning, daily stand-up, sprint retrospective, and planning game. For example, in a daily stand-up meeting agile teams clarify their cumulative work done by the team in a burn-down chart [S10]. In a sprint retrospective project knowledge is shared when agile teams share issues about lack of time to accomplish some tasks and uncompleted tasks in the last sprint which can lead to some changes in the timeline [S6,S11].

Process knowledge is managed in several agile practices, such as sprint/release planning, daily stand-up, sprint retrospective, pair programming and refactoring [S6,S11,S19,S24]. Process knowledge in these practices includes the knowledge about the system flows that are visualised in UML modeling or other visualisations that represent coding flow, features and business processes [S11].

With regards to knowledge management strategies applied, agile teams were seen to use: discussions (e.g. sharing requirements), artifacts (e.g. user stories) and visualisations (e.g. burn-down charts), to manage the project, product and process knowledge [5]. See summarized description in Table 2.

Table 2. Knowledge Management Strategies (KMS) in ASD

Knowledge Management Strategies (KMS)	***Description***	***Examples in ASD practices***
Discussions	Verbal communication that involves interaction among agile team members which aims to share knowledge	Sharing requirements [S20]; progress; plan and impediments [S13]; feedback; ideas/solutions [S24]; system flow; coding; techniques; design problems [S11]; coding problems; techniques; analysing; estimating and negotiate to agree; communication over video conference (e.g. Skype) [S47]
Artifacts	Physical forms that contain specific product features and project information	Story cards from Product backlog and Sprint backlog [S20]; user stories; task card [S46]; the card of code; code repositories [S13], JIRA [S47], Wiki [S6]
Visualisations	Strategies that clarify the resources about product, process and project into a visualised form.	Information radiators; UI prototyping [S13]; UML modelling [S11]; Burn down chart [S25].; story cards on the Scrum board; wall; action points [S46]; showing the code/working code [S5], JIRA, Wiki [S47], Microsoft Excel [S8]

Discussion was the most commonly used knowledge management strategy across all agile practices (e.g. release/sprint planning, daily stand-up and retrospective). This strategy facilitates agile teams to share knowledge, in particular process and project knowledge [S11,S13,S20,S24,S47]. Process knowledge was shared during discussions where agile teams share issues, ideas, solutions, new techniques in solving problems [S13] (e.g. coding, testing) and feedback, and negotiate with team members in making plans or decisions [S24]. Project knowledge was also included in the discussions where the content of the discussion related to the project, such as blockers in the last sprint which can affect the project timeline and other team members' progress [S11].

Artifacts in agile practices were commonly used to share product knowledge which included product requirements (e.g. in the form product backlog) and were further broken down into user stories. The product backlog also helped capture product knowledge through task cards for coding [S46], design and user interface development [S20].

Visualisation is the technique used to manage knowledge in a visible and accessible form. This strategy helped agile teams to support tacit knowledge sharing among team members. For example, the Scrum board was used to show progress based on the story cards; the whiteboard for information such as feedback, feelings and action points in the retrospective meetings [S46]; the burn-down chart for showing team progress, achievements and performance [S25] and software code on display screens for showing working code in pair programming [S48]. The most commonly used visualisation strategy in agile practices was the Scrum board, which contained all user stories and presented teamwork progress through the work status of team members [S11].

5 Discussion

In this section, we discuss a theoretical model of knowledge management in ASD which emerged from analyzing the knowledge types (KT), knowledge management strategies (KMS) and agile practices (AP) as described in the previous section.

Table 3. Agile practices that support knowledge management

Agile Practices (AP)		Knowledge Types (KT) Supported	Knowledge Management Strategies (KMS)	Supporting Primary Studies
Scrum Practices	Sprint/Release Planning	Product knowledge; Project knowledge; Process knowledge.	Discussions; Artifacts; Visualisations.	S11,S13,S14,S20,S24, S38-S40
	Daily Scrum	Product knowledge; Project knowledge; Process knowledge.	Discussion; Artifacts; Visualisations.	S6,S7,S10-S12,S23,S24, S41-S43
	Sprint Retrospective	Product knowledge; Project knowledge; Process knowledge.	Discussion; Artifacts; Visualisations	S11,S13,S16,S24,S44-S46
XP Practices	Pair Programming	Product knowledge; Process knowledge.	Discussion; Artifacts; Visualisations	S3,S5,S7,S17,S19,S22 ,S48
	Refactoring	Product knowledge; Process knowledge.	Discussion; Artifacts; Visualisations	S15,S17,S19,S25
	Planning game	Product knowledge; Project knowledge.	Discussions; Artifacts; Visualisations.	S16,S17,S19,S27

Table 3 presents a summarized view of the agile practices, knowledge types, and knowledge management strategies identified in this review. The first column lists the agile practices (AP) identified in the literature (section 4.1); the 'Knowledge Types (KT) Supported' column lists the knowledge types as related to agile practices (section 4.2 and Table 1); 'Knowledge Management Strategies (KMS)' column lists the knowledge management strategies inherent in the agile practices (section 4.2 and Table 2); and the primary studies, which can be seen in [5], supporting these findings. We found that all three types of knowledge were addressed in all the Scrum practices in varying degrees.

Furthermore, in synchronising teamwork through practices such as the daily stand-up and inspecting the process through retrospectives, agile teams combine product, project and process knowledge and build a framework to solve the problems or find ways to improve. Dingsøyr [14] explained that agile teams gather and link the knowledge through several processes referring to Nonaka and Takeuchi's knowledge creation theory. We expand on this work by highlighting the knowledge types and management strategies involved in these processes: Socialization, a process of sharing tacit knowledge (e.g. sharing mental models and technical skills) is achieved through discussion (identified as a KM strategy in our study). Externalization occurs when agile teams gain the shape of metaphors, concepts and models in written form, such as documentation, diagrams or artifacts (i.e. types of product and project knowledge). Agile teams gain and process the externalized knowledge to understand about "know-how" (i.e. a type of process knowledge) as part of the internalization process. The final process is a combination, where agile teams compile the knowledge from different sources (e.g. artifacts, meetings, board) in order to transform it into action.

Despite the inclusion of product, process and project knowledge in daily agile practices, some knowledge management related challenges were also identified in other reviews. Ringstad et.al. [15] and Stray et.al.[16] mention some challenges including: lack of focus on what was working well; no specific discussion about improving teamwork; and difficulty in transforming lessons learned into action [17].

The results of this review indicate that there is a discrepancy between Scrum theory and real practice, which could be attributed to the effectiveness of using knowledge in each practice. In Scrum practices (e.g. sprint/release planning meeting, daily stand-up and retrospective meeting) where product, process and project knowledge were involved, agile teams do not fully pay attention to the knowledge at the same degree, which means that they do not use the knowledge effectively. Another interesting finding from Scrum practice was that agile teams also tend to discuss product and project knowledge in the retrospective, which is theoretically meant to focus on the process alone.

In XP practices, product and process knowledge are discussed in pair programming and refactoring (see the description in Table 3). These findings are aligned with the theoretical aims of these practices. Pair programming aims to enhance agile team skills by working in pairs [1]. In practice, agile teams discuss coding issues, integrate with design and learn from other team member's skills. Similarly, refactoring focuses on maintaining coding and design, which involves product knowledge and process knowledge. In addition, Table 3 shows that the planning game in XP refers to product and project knowledge, being consistent with the theory [1] as this practice aims to

capture and analyse overall product requirements and build plans to accomplish the tasks.

The list of agile practices in Table 3 shows that most primary studies pay more attention to practices such as pair programming, daily Scrum and release/sprint planning meetings. Because meetings embody interaction and communication, these practices emphasize tacit knowledge sharing rather than explicit knowledge; however, there is evidence about some ways to transform tacit knowledge into explicit knowledge, such as storing it on sticky notes, paper or online documentation. Thus, important information or knowledge-related artifacts could be managed and used as a reference for team members.

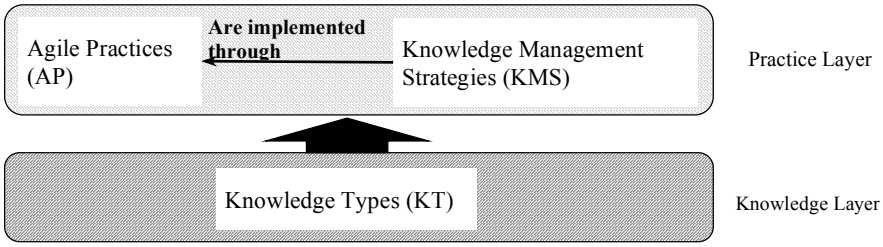

Fig. 4. A theoretical model of Knowledge Management in Agile Software Development

Based on the findings of this SLR, a theoretical model of knowledge management in ASD was developed. The theory consists of Knowledge Types (KT), Knowledge Management Strategies (KMS) and Agile Practices (AP). Fig. 4 depicts the theoretical model that illustrates two layers involved in knowledge management in ASD: a knowledge layer and a practice layer. The knowledge layer includes the three knowledge types and is a fundamental layer on which the practice layer functions. The practice layer includes agile practices and knowledge management strategies (or practices). The knowledge types from the knowledge layer are required and used in the agile practices and knowledge management strategies in the practice layer.

Our hypothesis is that the three knowledge types are managed by performing agile practices and knowledge management strategies, and the practices work using the knowledge types involved. For example, in daily stand-up (an agile practice), discussion, artifacts and visualisations (knowledge management strategies) are implemented to manage product, project and process knowledge (knowledge types). It can be seen that artifacts, visualisation and discussion in daily stand-up require particular knowledge to be managed. Without using artifacts on the scrum board and discussing the stories, the knowledge in daily stand-up cannot be managed effectively.

5.1 Implications

For practitioners, this SLR shows that there are three specific types of knowledge involved in everyday agile practice, which suggests that knowledge is necessary to be embedded and referred to by agile teams. It also seems important for agile teams to identify the three types of knowledge included in each knowledge management strategy (e.g. discussions, artifacts and visualisations), which must be applied by agile teams in

everyday practice. Thus, in order to gain benefits of knowledge management, we suggest that practitioners need to pay more attention to the types of knowledge described in this review, focus on how to manage that knowledge using the strategies discussed and implement the knowledge management concepts consciously in agile practices.

In terms of research, the SLR results suggest that instead of managing knowledge in ASD by classifying it as tacit and explicit knowledge, the specific explanation of knowledge management in ASD based on software engineering would be more relevant for agile teams, such as product, project and process knowledge involved in each agile practice. It also seems there is a need for further study into the knowledge types and management strategies involved in agile practices.

5.2 Limitations

We now discuss some of the limitations of this SLR. First is related to completeness. Despite our best efforts, and as common with most SLRs, there is a possibility that some articles published in some journals and conferences, which can address the research questions, were missed in this SLR. Thus, our results must be classified as applying to the papers selected from the three major digital libraries.

Furthermore, we are aware that some questions might arise about the selected articles. Thus, we defined inclusion and exclusion criteria (listed in [5] due to space constraints) as a protocol in selecting primary studies guided by the research questions. As well as some papers that described knowledge management theories in ASD, the implementation of knowledge management theories was reported but not discussed in detail in the primary studies, and are therefore not included in this SLR. We also assumed that selecting primary studies based on industrial empirical results would be more valuable for agile practitioners than summarizing findings from educational settings.

6 Conclusion

This systematic literature review set out to analyse and summarize the empirical research on knowledge management in agile software development (ASD). We analysed 48 primary studies filtered from an initial pool of 2317 papers using inclusion and exclusion criteria, presenting agile practices that support knowledge management in ASD.

The results of this SLR describe the knowledge types and knowledge management strategies in everyday agile practices. The most important contribution of this SLR is providing a new understanding of knowledge management in ASD that involves managing three different types of knowledge – process, project and product – by implementing three knowledge management strategies – discussions, artifacts and visualisations – during every day agile practices. Understanding these specific dimensions of knowledge and specific knowledge management strategies will help agile practitioners become aware of, and enable them to manage, the knowledge in everyday agile practices effectively.

Acknowledgement

This research is supported by the Indonesia Endowment Fund for Education (LPDP) S-669/LPDP/2013 as scholarship provider from the Ministry of Finance, Indonesia.

References

1. Beck, K.: Extreme Programming Explained: Embrace Change. Addison- Wesley Professional (1999)
2. Royce, W.: Managing The Development of Large Software Systems. IEEE WESCON, vol. 26, pp. 328-338. IEEE (1970)
3. Yanzer Cabral, A.R., Ribeiro, M.B., Noll, R.P.: Knowledge Management in Agile Software Projects: A Systematic Review. Journal of Information and Knowledge Management 13, (2014)
4. Neves, F.T., Rosa, V.N., Correia, A.M.R., Neto, M.d.C.: Knowledge Creation and Sharing in Software Development Teams Using Agile Methodologies: Key Insights Affecting Their Adoption. 6th Iberian Conference on Information Systems and Technologies (CISTI 2011) 1-6 (2011)
5. Andriyani, Y., Hoda, R., Amor, R.: Research Literature of Knowledge Management in Agile Software Development (ASD) – Technical Report. (2017)
6. Davenport, T.H., Prusak, L.: Working Knowledge-How Organizations Manage What They Know. Harvard Business School Press 5, 193-211 (1998)
7. Ikujirō, N., Takeuchi, H.: The Knowledge-Creating company : How Japanese Companies Create the Dynamics of Innovation. Oxford University Press 1995, New York (1995)
8. Ebert, C.D.M., J: Effectively utilizing project, product and process knowledge. Information and Software Technology 50(6), 579-594 (2008)
9. Lindvall, M., Rus, I.: Knowledge Management for Software Organizations. Managing Software Engineering 73-94 (2003)
10. Bjørnson, F.O., Dingsøyr, T.: Knowledge Management in Software Engineering: A Systematic Review of Studied Concepts, Findings and Research Methods Used. Information and Software Technology, vol. 50, pp. 1055-1068 (2008)
11. Kitchenham, B.A., Pfleeger, S.L., Pickard, L.M., Jones, P.W., Hoaglin, D.C., El Emam, K., Rosenberg, J.: Preliminary Guidelines for Empirical Research in Software Engineering. Software Engineering, IEEE Transactions on 28, 721-734 (2002)
12. Braun, V., Clarke, V.: Using thematic analysis in psychology. Qualitative research in psychology 3, 77-101 (2006)
13. Deemer, P., Benefield, G., Larman, C., Vodde, B.: A Lightweight Guide to the Theory and Practice of Scrum Version 2.0. vol. 2015, (2012)
14. Dingsoyr, T.: Value-based knowledge management: The contribution of group processes. Value-Based Software Engineering, pp. 309-325 (2006)
15. Ringstad, M.A., Dingsøyr, T., Moe, N.B.: Agile process improvement: Diagnosis and planning to improve teamwork. pp. 167-178. Springer (2011)
16. Stray, V.G., Moe, N.B., Dingsyør, T.: Challenges to teamwork: A multiple case study of two agile teams 12th International Conference on Agile Processes in Software Engineering and Extreme Programming: XP 2011 77 146-161 (2011)
17. Andriyani, Y., Hoda, R., Amor, R.: Reflection in Agile Retrospectives. International Conference on Agile Software Development, vol. 283, pp. 3-19. Springer, Cologne, Germany (2017)