Distributing Expertise in Agile Software Development Projects

Mawarny Md. Rejab
School of Engineering
and Computer Science
Victoria University of Wellington,
Wellington, New Zealand
Mawarny.Md.Rejab@ecs.vuw.ac.nz

James Noble
School of Engineering
and Computer Science
Victoria University of Wellington,
Wellington, New Zealand
kjx@ecs.vuw.ac.nz

George Allan
School of Engineering
and Computer Science
Victoria University of Wellington,
Wellington, New Zealand
george.allan@ecs.vuw.ac.nz

***Abstract*—The distribution of expertise in Agile teams is vital to enable team knowledge to be shared, preserved, and accessed when it is needed. Most studies emphasize knowledge sharing but a few empirical studies focus on skills. Integrating knowledge and skills is vital to leverage expertise in Agile teams. Moreover, it is not easy to leverage expertise by distributing expertise in Agile teams. Through a Grounded Theory study involving 18 Agile practitioners based in New Zealand and Australia, we revealed five approaches to distributing expertise in Agile teams: *embracing a master-apprentice model, coaching and mentoring, engaging hands-on learning, establishing discussion platforms* and *disseminating explicit knowledge*. Distributing expertise will provide insight into how Agile team members disseminate available expertise and pull new expertise into Agile teams.**

I. Introduction

The Agile team is a cross-functional team that includes all the expertise necessary for every phase involved in developing software [1]. All team members must understand each team member's role and participate beyond their area of expertise [2]. Agile team members do not just rely on their expertise in choosing tasks, but also tend to perform other tasks when needed. This is particularly important when Agile team members have to be involved in multiple projects at the same time. Relying on the same person for a variety of projects tends to cause bottlenecks. In order to avoid bottlenecks, there is a high tendency to absorb others' responsibilities. Thus, it is important to ensure the team knowledge can be shared, preserved, and accessed through distributing expertise.

Knowledge management is defined as *"the process of capturing, distributing, and effectively using knowledge"* [3]. This definition states that distribution of expertise is a part of knowledge management. Duhon [4] asserted that expertise constitutes knowledge management assets. Expertise refers to skills and knowledge that are retained by individuals to produce an outstanding performance [5] plus the ability of a person to generate knowledge in solving specific problems [6][7].

A number of researchers have studied knowledge sharing in the Agile software development context by emphasizing knowledge, rather than skills [8][9]. Integrating knowledge and skills is vital to leveraging expertise in Agile teams. Cho and his colleagues [10] posited that it is not easy to retain the distribution of expertise in Agile teams. Thus, this paper presents results from ongoing study addressing how Agile team members leverage expertise through distribution of expertise.

The rest of this paper is structured as follows: the second section describes Grounded Theory, our research methodology; the third section presents the research finding; the fourth section discusses our research findings; and the last section puts forward our conclusions.

II. Research Methodology

This study requires in-depth exploration of human behavior and social interaction from empirical data. As an inductive research method, Grounded Theory is used to infer new theories from observed data [11]. This study employs Grounded Theory to conceptualize and theorize about the underpinnings of distributing expertise from Agile software development perspectives.

A. Data Collection

Through Grounded Theory, we employ interviews as data collection methods. Interviews provide reliable data sources because the researcher has direct contact with participants during data collection [12]. This direct contact enables us to gain a deeper understanding of participants' concerns. As depicted in Table 1, semi-structured interviews have been carried out with 18 Agile practitioners from different software organizations based in New Zealand and Australia. This study requires a broad range of Agile roles to enable the triangulation of findings. Different roles provide different insights and perspectives toward distributing expertise in Agile teams. The data collection will continue until no new data emerges, which means theoretical saturation has been reached [11].

B. Data Analysis

A continuous interplay between data collection and analysis is the pivotal procedure in Grounded Theory [13]. In order to allow the emergence of theory, data analysis is done as soon as the first interview has been conducted [11]. Key point coding is used to analyze the interview transcripts in detail. We collate the key points by examining phrases, words, sentences from the interview transcripts [14]. Then, we construct codes by rephrasing key points with meaningful labels. In order to look for similarities and differences, constant comparison is used to compare every emerging code with the previous codes.

978-0-7695-5222-4/14 $31.00 © 2014 IEEE
DOI 10.1109/AGILE.2014.16

TABLE I. SUMMARY OF RESEARCH PARTICIPANTS AND AGILE SOFTWARE PROJECTS

Person	Location	Agile Role	Agile Methods
P1	New Zealand	Developer	XP and Scrum
P2	New Zealand	Agile Coach	XP, Scrum & Kanban
P3	Australia	Agile Consultant	Not specified
P4	New Zealand	Agile Coach	Scrum and XP
P5	New Zealand	Software Tester	Not specified
P6	Australia	Team leader	Not specified
P7	New Zealand	Agile Consultant	Scrum and XP
P8	Australia	Agile Coach	Scrum, XP, Kanban, Lean
P9	New Zealand	Business Analyst	Not specified
P10	New Zealand	Software Tester	Scrum
P11	New Zealand	Project Manager	Scrum
P12	New Zealand	Agile Coach	Scrum and Kanban
P13	New Zealand	Agile Coach	Scrum and Kanban
P14	New Zealand	Product Owner	Scrum
P15	New Zealand	Agile Coach	Scrum and Kanban
P16	New Zealand	Agile Coach	Scrum and Kanban
P17	New Zealand	Developer	Scrum
P18	New Zealand	Software Tester	Scrum

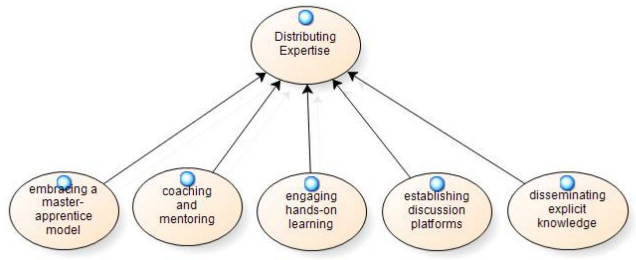
Distributing Expertise
embracing a master-apprentice model
coaching and mentoring
engaging hands-on learning
establishing discussion platforms
disseminating explicit knowledge

Fig. 1. The emergence of the category *"Distributing Expertise"* from underlying concepts

Similar codes with common themes are grouped together and emerge as a concept. Many concepts emerge, and constant comparison is repeated until concepts form a category. A category is a group of similar concepts that are used to generate the core category. To date, several categories have been emerged including *"distributing expertise"*.

III. RESEARCH FINDINGS

The category *"distributing expertise"* emerged from the data analysis to describe how Agile team members disseminate available expertise and pull new expertise into the team. This paper revealed five approaches to distributing expertise in Agile teams:*embracing a master-apprentice model, coaching and mentoring, engaging hands-on learning, establishing discussion platforms* and *disseminating explicit knowledge.* Figure 1 depicts the emergence of the category *"distributing expertise"* from underlying concepts.

A. Embracing a Master-Apprentice Model

Participants reported that master and apprentice relationships focus on pulling new expertise into Agile teams. A master is an internal or external expert who has responsibility to bring the new expertise into Agile teams, whereas the apprentice is someone seeking new expertise. Learning by doing together is the focal element in the master-apprentice model:

"The master is where you pull skills you don't have [sic]. Someone internal, or you, might bring contractors or a consultant for a small period of time. They start work together. The apprentice learns as well as he can and learns to become a master." - P3, Agile Consultant.

The master-apprentice relationship is a temporary scenario, whereby the master will leave the Agile team once the apprentice has grasped the new expertise. In order to disseminate the expertise widely into Agile teams, the same implementation is repeated by rotating the apprenticeship with other team members. At this point, the apprentice becomes a master and trains others based on what he or she has learnt:

"When the expert goes, the team [still] has knowledge of how to do the piece of work." - P3, Agile Consultant.

Developing a learning organization is a benefit gained from the master-apprentice model. A learning organization creates a culture that encourages and facilitates continuous employee learning. Through a learning organization, Agile team members tend to transfer from individual learning to shared learning.

B. Coaching and Mentoring

Participants stated that coaching and mentoring are other methods to obtain the required knowledge and skills:

"If the team doesn't know anything about the skill, the best thing is to invite a mentor [or coach] from the real community and teach us [sic]." - P2, Agile Coach.

Agile teams adopt mentoring and coaching through Agile coach roles. An Agile coach plays an important role in facilitating team members to gain new knowledge and skills. An Agile coach can also act as a mentor in strengthening team members' skills, particularly for those who have failed to meet expectations. With the proper guidance and adequate expertise, team members can gain benefits through the coaching and mentoring approach:

"Bring the coach for several times, and pair and teach [them] how to do [sic]."- P6, Team Leader.

The findings of this study indicate that there are a few differences between the coaching and mentoring approach and the master-apprentice model. While both approaches aim to disseminate expertise, the master-apprentice model puts a high priority on producing an expert or a master in the specific skill. In terms of implementation, the coaching and mentoring focuses on facilitating and training, whereas the master-apprentice emphasizes learning by collaborating. Both approaches complement one another and are essential to the successful distribution of expertise in Agile teams.

C. Engaging Hands-on Learning

Transforming the learning experience through hands-on exercises provides opportunity to develop expertise, as well as distribute expertise. Drawing from the research findings, there are three ways to engage hands-on learning in Agile teams: coding dojo, pair-programming, and internship programs.

A coding dojo involves two team members working together to solve the programming problem with feedback and guidance from the audience. Role rotation is essential to enable other audience members to have hands-on programming within the limited time. A coding dojo provides a space for Agile team members to learn, practise, and share their programming skills:

"The developers run workshops, [by] running a coding Dojo. Then, they will be paired on the computer to tackle some problems. We rotated the team around and we can see what other people are doing." - P15, Agile Coach.

Pair-programming is embedded in coding dojo practices. Most participants affirmed that pair-programming is a powerful practice in disseminating their expertise to other team members:

"Learning from one another through pair programming. Keep pairing and everybody will learn and pull expertise." - P2, Agile Coach.

Hands-on learning can also be gained through the internship program. This program requires Agile team members to attach themselves with other teams for a short period of time. The goal of the internship program is to bring the new expertise into Agile teams:

"Internships are like short apprenticeships where people can learn from people who are really good at something by working alongside them. They temporarily join another squad for 2 - 4 weeks and learn as much as they can." - P12, Agile Coach.

In certain circumstances, these hands-on learning methods are incorporated in a master and apprentice model, as well as coaching and mentoring. For instance, pair-programming is applied in the master-apprentice model. However, most identified hands-on learning methods are applicable to foster shared learning even without master or coach involvement.

D. Establishing Discussion Platforms

Having meaningful discussion through the right platform tends to enable the distribution of expertise in Agile teams. Three discussion platforms have been identified from this research finding: interest groups, chatting tools, and robust debate.

Our finding reveals that interest groups provide a solid base of interconnection among peers who have similar interests in a particular area of expertise. Agile team members meet for face-to-face discussions on a regular basis to share their expertise and learn from each other:

"They set up some special interest groups for sharing knowledge within the company such as testing community, and developer community." - P15, Agile Coach.

A chatting tool is another discussion platform that facilitates the distribution of expertise. Agile team members have the ability to communicate by sharing their expertise at any time:

"We got Internet Relay Chat (IRC). It is an in-house chat [tool] for chatting within a specific group such as testers group, developers group, with different channels. This place is where we can share the information." - P17, Software Tester.

One of the participants noted that it is normal to have robust debates when they were collaborating:

"We worked on the story. We had quite a lot of debate. If you look [from] outside, it might look like we are arguing and screaming at each other. But look inside, it is just a reflection of passion. All the time the idea is moved." - P4, Agile Coach.

The aim of robust debate is to reach a resolution by forming arguments in the right way. From the positive insight, a robust debate encourages Agile team members to stimulate their minds in order to generate ideas. Open and genuine debates allow the distribution of expertise particularly in solving problems and making decisions.

E. Disseminating Explicit Knowledge

Most concepts emphasize distributing tacit knowledge. Another concept *"disseminating explicit knowledge"* , however emerged from our findings. This concept describes how explicit knowledge can be disseminated, shared, and preserved in Agile teams through video, sketching on whiteboards, and document management tools.

A video is the preferred method for delivering knowledge and skills among Agile team members. The video image movement is a good mechanism to present new software skills. Replaying the same video contents for different audiences tends to save time and cost:

"You just point the video and let them watch the video. If you have 100 people, then 100 people can watch the video. Everybody can have the same understanding [sic]."- P2, Agile Coach.

Sketching on whiteboards facilitates Agile team members to externalize their mental models of expertise. The tacit knowledge can be visualized in explicit form through the sketch. The sketch has the ability to support the face-to-face communication in strengthening the distribution of expertise:

"That's very much like having some design sketch on a whiteboard. We need the sketch to understand and communicate how to do the code [sic]." - P2, Agile Coach.

Two document management tools have been identified from our findings: Google Docs and Wikis. Google Docs allows Agile team members to create, edit, and share documents online. One participant claimed that he shared documents not just for sharing the software project information, but also for disseminating individual knowledge and skills. Everyone has the opportunity to access and update the shared documents in order to distribute expertise:

"For us as testers, we have spreadsheets on Google Docs for [saving] command lines. We share the command lines that we know on the spreadsheets, which are accessible for everyone." - P17, Software tester.

A strategy used by one of the participants for orientation purposes was sharing tribal knowledge through wikis:

"Now, many teams use wikis. Wikis actually report on some tribal knowledge...if the new developer joins, then they use this induction [sic]." - P2, Agile Coach.

Tribal knowledge is undocumented knowledge that should be known by Agile team members. Wikis allow the transition of tribal knowledge, from undocumented knowledge to written knowledge. Through Wikis, Agile team members can share information that should be known by a newcomer before they engage in software development projects. Wikis play an important role in preserving tribal knowledge and supporting organizational memory.

IV. Discussion

A coaching and mentoring approach is a common method to share tacit knowledge in Agile teams [15][16]. Several research studies have demonstrated the success of coaching and mentoring in sharing knowledge in Agile teams [17][18]. In contrast, there is a paucity of studies that focus on the master-apprentice model [19]. Consequently, more findings are needed to clearly define the proper implementation of a master-apprentice model specifically in the Agile software development context.

A majority of participants asserted that pair-programming was their preference in sharing knowledge among peers. Thus, pair-programming are incorporated with other practices such as coding dojo, coaching, and master-apprentice models [20][21][22]. In terms of internships, Lindvall et al.'s study [17] mentioned the internships program in Agile teams. However, there is limited discussion on internships in the Agile software development context. Thus, further investigation is needed to clearly define how internships are implemented in Agile software development.

This study also indicated that distributing expertise can be established through discussion platforms such as robust debate. Our finding contradicts a study completed by Chau et al.[8], which posited that debate tends to delay the knowledge transfer in Agile teams. Proper implementation to facilitate debates, however, is essential to ensure that the knowledge sharing between individuals takes place successfully [23]. Our finding is consistent with Dessai et. al's study [24], which indicated that chatting tools assist the understanding of others' expertise and also facilitate the sharing of expertise among Agile team members [24]. Future data collection and analysis will reveal more discussion platforms that facilitate the distribution of expertise in Agile teams.

Wikis gain positive attention among Agile practitioners, serving as knowledge platforms for the Agile software development community to organize, share, integrate, and preserve explicit knowledge effectively [8][25]. Wikis can help in distributing expertise by providing each team member with a flexible privilege to access, update, and share their explicit knowledge. According to Settina and Heijstek [26], further investigation is needed to explore the adoption of wikis and Google Docs in Agile teams.

V. Conclusion

This paper describes several techniques for Agile teams to disseminate the expertise and pull new expertise into teams. The distribution of expertise in Agile teams is vital to enable team knowledge to be shared, preserved, and accessed when it is needed. Distributing expertise ultimately leads to successful cross functional teams. Besides the identified approaches, we believe that there are more approaches used by Agile practitioners to distribute expertise in Agile teams. Further data collection and analysis will divulge other approaches and when to choose the right approach in distributing expertise. In order to optimize the effectiveness of expertise distribution, we intend to investigate factors that affect the distribution of expertise in Agile teams.

Acknowledgment

We would like to express the deepest appreciation to Student Learning Support Service (SLSS) staffs and Dr. Siva Dorairaj for outstanding editing and proofreading

References

[1] J. Sutherland, "Scrum handbook," 2010. [Online]. Available: http://jeffsutherland. com/scrumhandbook. pdf/

[2] R. Hoda, "Self-organizing agile teams: A grounded theory," *Phd Thesis, Victoria University of Wellington*, 2011.

[3] T. H. Davenport, "Saving it's soul: Human-centered information management." *Harvard business review*, vol. 72, no. 2, pp. 119–31, 1994.

[4] B. Duhon, "It's all in our heads," *Inform*, vol. 12, no. 8, pp. 8–13, 1998.

[5] A. Mockus and J. D. Herbsleb, "Expertise browser: a quantitative approach to identifying expertise," in *Proceedings of the 24th international conference on software engineering*, 2002, pp. 503–512.

[6] K. Kuchinke, "The status of the theory and the literature," *Performance Improvement Quarterly*, vol. 10, pp. 72–86, 1997.

[7] R. Herling, "Operational definitions of expertise and competence," *Advances in developing human resources*, vol. 2, no. 1, pp. 8–21, 2000.

[8] T. Chau and F. Maurer, "Knowledge sharing in agile software teams," in *Logic versus approximation*. Springer, 2004, pp. 173–183.

[9] C. R. D. Souza, "Fostering inter-team knowledge sharing effectiveness in agile software development," 2012.

[10] J. Cho, R. Huff, and D. Olsen, "Management guidelines for scrum agile software development process," *Issues in Information Systems*, vol. 12, no. 1, pp. 213–223, 2011.

[11] B. G. Glaser and A. L. Strauss, *The discovery of grounded theory: Strategies for qualitative research*. Aldine de Gruyter, 1967.

[12] K. Charmaz, *Constructing grounded theory: A practical guide through qualitative analysis*. Sage Publications Limited, 2006.

[13] J. M. Corbin and A. Strauss, "Grounded theory research: Procedures, canons, and evaluative criteria," *Qualitative sociology*, vol. 13, no. 1, pp. 3–21, 1990.

[14] G. Allan, "A critique of using grounded theory as a research method," *Electronic Journal of Business Research Methods*, vol. 2, no. 1, pp. 1–10, 2003.

[15] S. Ryan and R. V. O'Connor, "Development of a team measure for tacit knowledge in software development teams," vol. 82, no. 2. Elsevier, 2009, pp. 229–240.

[16] A. Elshamy and A. Elssamadisy, "Divide after you conquer: an agile software development practice for large projects," pp. 164–168, 2006.

[17] M. Lindvall, V. Basili, B. Boehm, P. Costa, K. Dangle, F. Shull, R. Tesoriero, L. Williams, and M. Zelkowitz, "Empirical findings in agile methods," pp. 197–207, 2002.

[18] S. C. Misra, V. Kumar, and U. Kumar, "Identifying some important success factors in adopting agile software development practices," *Journal of Systems and Software*, vol. 82, no. 11, pp. 1869–1890, 2009.

[19] K. H. Judy, "Agile principles and ethical conduct," in *System Sciences, 2009. HICSS'09. 42nd Hawaii International Conference on*. IEEE, 2009, pp. 1–8.

[20] D. T. Sato, H. Corbucci, and M. V. Bravo, "Coding dojo: An environment for learning and sharing agile practices," in *Agile, 2008. AGILE'08. Conference*. IEEE, 2008, pp. 459–464.

[21] V. A. Santos, A. Goldman, and C. D. Santos, "Uncovering steady advances for an extreme programming course." Centro Latinoamericano de Estudios en Informtica, 2012, vol. 15, no. 1, pp. 2–2.

[22] J. Evnin and M. Pries, "Are you sure? really? a contextual approach to agile user research," in *Agile, 2008. AGILE'08. Conference*. IEEE, 2008, pp. 537–542.

[23] S. Fernie, S. D. Green, S. J. Weller, and R. Newcombe, "Knowledge sharing: context, confusion and controversy," *International Journal of Project Management*, vol. 21, no. 3, pp. 177–187, 2003.

[24] K. Dessai and M. Kamat, "Application of social media for tracking knowledge in agile software projects," *Available at SSRN 2018845*, 2012.

[25] J. García, A. Amescua, M.-I. Sánchez, and L. Bermón, "Design guidelines for software processes knowledge repository development," *Information and Software Technology*, vol. 53, no. 8, pp. 834–850, 2011.

[26] C. J. Stettina and W. Heijstek, "Necessary and neglected?: an empirical study of internal documentation in agile software development teams," in *Proceedings of the 29th ACM international conference on Design of communication*. ACM, 2011, pp. 159–166.