# Knowledge Management in Practice: The Case of Agile Software Development

Meira Levy
*Deutche Telekom Laboratories@Ben Gurion University, and*
*Department of Industrial Engineering & Management*
*Ben-Gurion University of the Negev, Israel*
*lmeira@bgu.ac.il*

Orit Hazzan
*Department of Education in Technology and Science*
*Technion - Israel Institute of Technology*

## Abstract

*Knowledge is considered as the main competitive asset of the organization. One of the knowledge management (KM) cornerstones is improving productivity by effective knowledge sharing and transfer. However, from the game theory perspective, the main constraint is that people tend not to collaborate in uncertainty conditions, when collaborative behavior is not guaranteed, and sharing knowledge is time- and effort-consuming. Therefore, KM must be a practical aspect of the general organizational culture. Specifically, software development is a knowledge-intensive activity and its success depends heavily on the developers' knowledge and experience. In this presentation we highlight how the agile approach initiates a culture change that is in line with the culture change needed for a KM initiative. We discuss KM enablers that are embedded in the agile software engineering approach, and illustrate how collaborating processes and knowledge transparency can weaken the dilemmas people face and lead to better knowledge extraction and sharing.*

## 1. Introduction

Software development is a human-based knowledge-intensive activity where the level of uncertainty is high. There are many challenges concerning schedule, cost estimation, reliability, defects and performance due to great increases in software complexity and quality demands. Furthermore, high staff turnover, psychology and sociology dynamics between team members, and volatile software requirements, are only a few examples of the challenges of software projects. Thus, the success of a software project depends heavily on the knowledge and experience brought to the project by its practitioners [2].

Knowledge is considered as the main competitive asset of the organization, enabling the enterprise to be productive and to deliver competitive products and services. There are several knowledge frameworks that describe knowledge-related practices. Holsapple and Singh [12] describe potential sources of competitive advantages in a firm. They developed the knowledge chain model that identifies five primary knowledge manipulation activities and four secondary activities. The primary activities include: knowledge acquisition, knowledge selection, knowledge generation, knowledge internalization and knowledge externalization; the secondary activities include: knowledge leadership, knowledge coordination, knowledge control, and knowledge management (KM). O'Dell and Grayson [17] illustrate knowledge transfer comprehensive framework that include the knowledge life-cycle practices (create, identify, collect, organize, share, adapt and use) and the environment – cultural and structural – necessary for dynamic and successful KM processes

The organizational knowledge is embedded in people, systems, procedures and products. Knowledge workers are required to improve their work on a daily basis in a process that cumulates into a significant improvement in performance for the entire enterprise [6][21]. One of the cornerstones of KM is improving productivity by effective sharing and transfer of knowledge, which tends to be time-consuming and often impossible [21].

Nonaka [16] distinguishes between implicit (tacit) and explicit knowledge. Explicit knowledge is stored in textbooks, software products and documents; implicit knowledge is stored in the minds of people in the form of memory, skills, experience, education, imagination and creativity. Spender [18] classifies knowledge in terms of implicit, explicit, individual and collective knowledge. There is a common understanding that both implicit and explicit are important; however, implicit knowledge is more difficult to identify and manage [2]. Alavi and Leidner [1] propose that knowledge represents information possessed in the minds of individuals, specifically "personalized information (which may or may not be new, unique, useful, or accurate) related to facts, procedures, concepts, interpretations, ideas, observations, and judgments" (p. 109). They posit that information is converted to knowledge once it is processed in the mind of individuals and knowledge becomes information once it is articulated and presented in the form of text, software product or other means. The receiver can then cognitively process the information so that it is converted back into tacit knowledge.

According to Fahy and Prusak [8], tacit knowledge consists of perspectives, perceptions, beliefs, and values and has a central role in capturing and assimilating explicit knowledge. Many companies invest in explicit KM, supporting KM initiatives that focus mainly on technological solutions (e.g., content management). Yet, rather than fostering prosperous sharing communities, many KM initiatives end up being passive and "unintelligent" systems [21] One of the greatest challenges of knowledge organizations is to extract tacit knowledge possessed by their knowledge employees, so as to apply the right knowledge at the right place when needed and to encourage innovation. One of the main barriers to these efforts is managers' fear and misunderstanding of the nature of tacit knowledge, its values, and importance [8]. Another related barrier to KM initiatives is the separation between the work processes and the KM processes, and the focus on meeting deadline requirements that causes KM tasks to be avoided. Davenport and Prusak [4] say that "Everyone today is already too busy; we don't have time to add KM on top of our jobs. Expecting knowledge workers to peruse repositories of lessons and experiences in their spare time, or to share their own learning's at leisure, is highly unrealistic". (p. x)

In this paper, we explore how agile development environments can be seen as a platform for the extraction of tacit knowledge without extra effort, overcoming cultural and psychological barriers.

## 2. Cultural Aspects of Knowledge Management

In addition to the work constraints, from the game-theory perspective (specifically the prisoner and the public-goods dilemmas), we know that people tend not to cooperate in environments in which the behaviors of processes and colleagues are not transparent [3][9]. In addition, the time and effort required for knowledge extraction prevent the knowledge workers from contributing to the public domain.

Gold et al. [9] suggest four knowledge processes: acquisition, conversion, application, and protection, in addition to three KM infrastructure capabilities: organization's technology, structure, and culture. These researchers found that both KM infrastructure capabilities and knowledge processes positively influence organizational effectiveness. Alavi and Leidner [1] ask which cultures foster knowledge creation and sharing and whether cultural change should occur before KM initiatives can be successfully undertaken or whether KM initiatives facilitate cultural change.

A knowledge-friendly organizational culture has been identified as one of the most important conditions leading to the success of KM initiatives in organizations [4]. Alavi and Leidner [1] claim that there are cultural barriers to KM that prevent employees from making knowledge available, sharing it with others, teaching and mentoring others, using their expertise to innovate, or finding ways of working smarter. Moreover, in many organizations, members feel that their promotion depends upon the expertise they generate and not on the extent to which they help others. Another barrier is that people may not realize what aspects of their knowledge are relevant for others. Without a systematic routine for knowledge capturing, a firm might not benefit from its accumulative tacit knowledge. In many organizations, a major cultural shift is required to change employees' attitudes and behavior so that they willingly and consistently share their knowledge and insights. In our work we attempt to explore how agile software development creates a cultural infrastructure for KM processes and practices.

## 3. Knowledge Management in Software Engineering

The main challenge of KM is to transfer implicit knowledge to explicit knowledge, as well as to transfer explicit knowledge from individuals to groups within the organization. Software developers possess highly valuable knowledge relating to product development, the software development process, project management and technologies. The software development work requires various forms of explicit as well as implicit knowledge. This knowledge is dynamic and evolves with technology, organizational culture, and the changing needs of the organization's software development practices [2]. Software development can be improved by recognizing the related knowledge content and structure as well as the required knowledge, and performing planning activities.

KM is comprised of the elicitation, packaging and management, and reuse of knowledge in all of its different forms, and in particular, software engineering artifacts as code, design, requirements, models, data, standards, and lessons learned. Thus, developing effective ways of managing software knowledge is of interest to software developers. However, it is not well understood how to implement this vision, as Aurum *et al.* [2] claim:

"The blind population of knowledge repositories will not lead to success. Rather, the careful and goal-oriented inclusion and packaging of knowledge for specific reuse scenarios should be aimed for. The 'store and hope for reuse' paradigm has failed in the past in its attempts to get code reused; it also will fail in the attempt to get comprehensive knowledge reused" (ibid, p. v).

## 4. Agile Software Engineering Approach

Agile software development methodologies have recently generated great enthusiasm among both practitioners and researchers [14][20]. Popular agile methods include Extreme Programming, Feature-Driven Development, Crystal Methods, Scrum, Dynamic Systems Development, and Adaptive Software Development. The common principle underlying agile methods emphasizes cooperative software development. The focus is more on people and on the dynamics of their interactions, rather than on elaborate requirements planning and rigid software development processes. A key concept promoted by the Agile Manifesto (see Table 1) is that people (software developers, customers and users) form the cornerstone of the software development process. The Agile Manifesto was formulated in 2001 by a group of 17 leading software practitioners and its principles are accepted by the community of agile development, no matter what specific agile method one uses.

**Table 1. The Agile Manifesto (http://agilemanifesto.org/)**

| |
|---|
| **Manifesto for Agile Software Development**<br>We are uncovering better ways of developing<br>software by doing it and helping others do it.<br>Through this work we have come to value:<br>Individuals and interactions over processes and tools<br>Working software over comprehensive documentation<br>Customer collaboration over contract negotiation<br>Responding to change over following a plan<br>That is, while there is value in the items on<br>the right, we value the items on the left more. |

Data indicate that agile software projects cope successfully with common problems of software projects. For example, according to the "State of Agile Development" survey[1] conducted by Version One and the Agile Alliance in 2007, 60% of respondents estimated a 25% or greater improvement in time-to-market; 86% estimated a 10% or greater improvement in time-to-market; 55% reported a 25% or greater improvement in productivity; 87% reported a 10% or greater improvement in productivity; 55% reported a 25% or greater reduction in software defects; and 86% of the respondents reported a 10% or greater reduction in software defects. Regarding value actually realized from the agile approach, 92% reported improvement in managing changing priorities, 74% reported improved team morale, and 66% reported improvement in business/IT alignment.

In our work we aim at understanding, whether, and if yes – how, KM practices contribute to these positive achievements of software projects. Specifically, we aim at identifying agile practices that foster KM in agile projects and characterizing in what ways they foster knowledge sharing and management.

This line of thought stands in line with Takeuchi and Nonaka [19] argument that that in today's fast-paces, fiercely competitive world, speed and flexibility are essential. Leading companies show specific characteristics in managing their new product development process among which are "Multilearning" and organizational transfer of learning. "Multilearning" refers to two dimensions of learning: across multiple levels (individual, group, and corporate) and across multiple functions. Transfer of learning occurs through "osmosis", when the knowledge people are assigned to other projects and by converting successful specific project activities to standard practice. However, external changes should trigger process redefinition, since standard practices, which are based on past good experience, are reexamined and improved in tune with the new realities of the outside environment. Scrum, one of predominant agile software development methods, for example, was developed based on this perspective.

## 5. Knowledge Management in Agile Software Engineering

In this section we theoretically explore how, agile software practices enhance the participants' awareness to different situations that lead to the extraction of tacit knowledge and to effective use of this knowledge. Based on Hazzan and Dubinsky [11] we briefly mention several practices we aim to explore with respect to KM aspects:

**Whole team.** The practice of *Whole Team* means that the development team (including all role holders and the customer) communicate in a face-to-face fashion as much as possible. The *whole team* practice is applied in several ways. First, the development team is co-located in a *collaborative workspace* – a space which supports and facilitates communication. Second, all team members participate in all the product presentations to the customer, hear the customer requirements and are active in the actual process planning. Third, role holders, that traditionally belong to separate teams (e.g., testers and designers), are integrated into the development team and process.

**Roles.** Within the *whole team* concept, each team member has an additional role, in addition to being a software practitioner. The rationale for this role distribution is that one person, usually, the team leader, no matter how skilled he or she is, can not handle in a professional manner all of the essential and complex responsibilities involved in software development projects. The distribution of responsibilities in the form of roles helps to control and manage these responsibilities. In addition, such a role scheme implies that all team members are involved in all parts of the developed software, each from his or her role perspective. See also Dubinsky and Hazzan [7].

**Collaborative workspace.** The walls of the development workspace serve as a communication means, constituting an informative and collaborative space. The information posted on the walls includes, among additional relevant information, the status of the personal tasks that belong to the current iteration and the measures taken. Thus, all project stake holders can be updated at a glance at any time about the project progress and status.

**Stand-up meetings**. The entire team holds daily *stand-up meeting* which usually take place in the morning for 10-15 minutes. In these meetings, each team member presents in 2-3 sentences the status of his or her development tasks and what he or she plans to accomplish during the day to come, both with respect to the development tasks and the personal role. When needed, one sentence can be added by each team member at his or her turn with respect to anticipated problems.

**Measures.** In order to navigate the development process so that a high quality product is developed,

[1] Agile Development: Results Delivered: http://www.versionone.net/pdf/AgileDevelopment_ResultsDelivered.pdf

agile software development processes are accompanied with measures on which all the project stake holder decide according to their needs. The importance attributed to measures is expressed by agile software development processes, among other means, by the special role – a *tracker* – assigned to one team member, who is in charge of the measures. Measures enable the development team to improve the development process, and consequently, the developed software. Measures also convey the message that the development process should be monitored and that this monitoring should be transparent and known to all project stake holders who participate in the development process.

**Customer collaboration.** Agile software development methods welcome the product customer to become part of the development process. The target is to get an on going feedback from the customers and to move on according to their needs. Such an on-going interaction avoids the need to speculate the customers' needs, which may lead to incorrect working assumptions. This practice implies that in agile software development all team members have access to the customer during the entire development process. This direct communication channel increases the chances that the software requirements are communicated correctly and it helps teammates cope successfully with change introduction at later stages.

**Pair programming.** This practice means that each code is developed by two teammates who together sit in front of the computer screen and in an interactive, communication-based process develop a specific development task [22]. It is important to note that even though the development is carried out in pairs, there is personal responsibility for each development task. During pair programming processes it is harder to be distracted and, hence, pairs tend to remain focused on the development task. In addition, each task is perceived on two levels of abstraction: of the driver – the one who works with the keyboard and thus thinks on a lower level of abstraction, and of the navigator – the mate who thinks about the development task on a higher level of abstraction. The application of the *pair programming* practice implies that all team members become familiar with all parts of the developed software and improve their comprehension of the entire software. This fact enhances a culture construction that fosters knowledge sharing.

## 6. Future Research

In our future research we aim at identifying specific elements of KM in the work process of agile teams. Specifically, the purpose of the research is to formulate KM practices embedded in agile software development in order to elevate the awareness of the agile community to benefits they may exploit from practices which are already integrated in their development process. This assertion is based on our working assumption that there are additional in-practice KM practices, which are fostered by agile software development, but have not yet formally outlined.

From this research target, we derived the following research questions:

1. What KM-oriented practices do agile teams use? How do these practices foster KM?
2. To what degree are these practices conceived by agile teams as KM-oriented practices?
3. How is the organizational culture related to agile KM-related practices?

To answer these questions, our research plan includes:

1. Observations in software houses in which teams employ an agile process;
2. Interviews with software practitioners about their conception of KM practices.

## 7. Conclusion

This paper is based on the assumption that KM is a vital part of any project in general and for software projects in particular. Specifically, our work aims to examine the integration of KM and agile software development which are two organizational processes that face common barriers when introduced and applied. The pairing of KM and agile software development is not new; a connection between the two concepts has been acknowledged by various researchers. For related discussions, see, for example, Dove [5], Holz *et al.*[13] and Levy and Hazzan [15]. This connection, however, is not surprising since both disciplines deal with organizational culture and change management. We suggest that our work will enable deepening the understanding of the connections and interconnection between these two processes and thus fostering relevant organizational processes.

## 8. References

[1] M. Alavi, M. and Leidner, D. E., “Review: Knowledge management and knowledge management systems: Conceptual foundations and research issues”, *MIS Quarterly*, 25(1), 2001, pp. 107-136.

[2] A. Aurum, R. Jeffery, C. Wohlin, and M. Handzic, (Ed.), *Managing Software Engineering Knowledge,* Springer-Verlag, New York, 2003.

[3] U. Cress, and S. Martin, "Knowledge sharing and rewards: A game-theoretical perspective". *Knowledge Management Research & Practice*, 4, 2006, pp. 283-292.

[4] T. H. Davenport and L. Prusak, *Working Knowledge*, paperback edition, Harvard Business School Press, Boston, 1998.

[5] R. Dove, "Knowledge Management, Response Ability, and the Agile Enterprise". *Journal of Knowledge Management*, 1999, pp. 18-35.

[6] P. Druker, *Post-Capitalism Society*, UK, Oxford, Butherworth-Heinemann, 1993.

[7] Y. Dubinsky, and O. Hazzan, "Using a roles scheme to derive software project metrics, QUantitative TEchniques for SoftWare" Agile Processes workshop, *Proceedings (and selected for the Post-Proceedings) of SIGSOFT 2004*, Newport Beach, CA, USA, 2004.

[8] L. Fahy and L. Prusak, "The eleven deadliest sins of knowledge management", *California Management Review*, 40(3), 1998, pp. 265-277.

[9] A. H. Gold, A. Malhotra, and A. H. Segars, "Knowledge management: An organizational capabilities perspective", *Journal of Management Information Systems*, 18(1), 2001, pp. 185-214.

[10] O. Hazzan, and Y. Dubinsky, "Social perspective of software development methods: The case of the prisoner dilemma and extreme programming", *Proceeding of the 6th International Conference on Extreme Programming and Agile Processes in Software Engineering*, Sheffield, UK, 2005, pp. 74-81.

[11] O. Hazzan, and Y. Dubinksy, "Agile Software Engineering, Undergraduate Topics in Computer Science" (UTiCS) Series, Springer, 2008.

[12] C. W. Holsapple and M. Singh, "The Knowledge Chain Model: Activities for Competitiveness", in Handbook on Knowledge Management, C. W. Holsapple, (Ed.), 2(43), Springer, Lexington KY, USA, 2003, pp. 215-251.

[13] H. Holz, and G. Melnik, "Research on Learning Software Organizations - Past, Present and Future, Advances in Learning Software Organizations", 6th International Workshop, LSO, 2004, pp.1-6.

[14] C. Larman, Agile and Iterative Development: A Manager's Guide, Addison-Wesley Professional, 2004.

[15] M. Levy and O. Hazzan, "Agile knowledge management", in the Encyclopedia of Information Science and Technology, Second edition, Khosrow-Pour, M. (Ed,), IGI Global, Hershy, PA, 2008, pp. 112-117.

[16] I. Nonaka, "A Dynamic Theory of Organizational Knowledge Creation", Organization Sciences, 5(1), 1986, pp. 14-37.

[17] C. O'Dell, and J. C. Grayson, "Identifying and Transferring Internal Best Practices", in Handbook on Knowledge Management, C. W. Holsapple (Ed.) 1(31), Springer, Lexington KY, USA, 2003, pp. 601-622.

[18] J. C. Spender, "Pluralist Epistemology and the Knowledge-Based Theory of the Firm", Organization, 5(2), 1998, pp. 233-256.

[19] H. Takeuchi and I. Nonaka, "The New Product Development Game", Harvard Business Review, 64 (January-February), 1986, pp. 137-146.

[20] D. Turk, R. France and B. Rumpe, "Assumptions underlying agile development processes", Journal of database management, 16(4), 2005, pp. 62-87.

[21] K. M. Wiig, and A. Jooste. "Exploiting Knowledge for Productivity Gains", in *Handbook on Knowledge Management*, C. W. Holsapple (Ed*.)* 2(45), Springer, KY, 2003, pp. 289-308.

[22] L. Williams, R. Kessler, R, W. Cunningham and R. Jeffries, "Strengthening the Case for Pair-programming", *IEEE Software*, July/August 2000, pp. 19-25.