Aspects of Knowledge Transfer in eXtreme Programming

Jaana Nyfjord, Stockholm University/Royal Institute of Technology, Forum 100, SE-164 40 Kista, Sweden; E-mail: jaana@dsv.su.se

ABSTRACT

The reason why software projects remain vulnerable to failure is essentially based on a knowledge management (KM) problem. One observation, however, is that the research in this area assumes, or is restricted to, traditional software development approaches. By excluding newer ways of developing software, such as agile software development, important aspects of KM in software engineering have been omitted. In this paper, a theoretical analysis of extreme Programming (XP) from a KM perspective is presented. The aim is to gain understanding of how the two fields relate but also to investigate to what extent aspects of KM can be beneficial to XP. The result shows that XP is more aligned with KM than expected. More importantly, the result suggests that a creative approach to KM can generate improved efficiency, higher productivity and higher quality in multi-team XP projects. A framework that can be used to analyse how XP supports KM in multi-team settings is also proposed.

1. INTRODUCTION

The failure and cancellation rate of software projects around the globe is still alarming. A significant number of the classic problems of delayed projects, cost overruns and meeting objectives keep being reported in surveys by for example the Cutter Consortium [10] and the Standish Group [20]. The question is why these problems remain, despite well-established knowledge about them.

Backlund [2] explains that the reason to why software projects remain vulnerable to failure is essentially based on a knowledge management (KM) problem. Organisations fail to learn from their own experiences. This is also supported by for example Ye and Fay [21], who argue that many software projects fail particularly because of the lack of knowledge transfer between the members of software project teams.

According to Backlund [2], the solution to this problem is to encourage software developers to learn from their own and others' experiences and to use this knowledge to change their development practices. Iivari [13], Ye and Fay [21] and Wastell [22] go even further and suggest that the software development process should be viewed as an ongoing learning process engaging both domain specialists and software professionals. There is plenty of research arguing for the importance of KM in software development activities. Much of the work in software engineering carried out over the last twenty years should confirm this fact [3]. One observation, however, is that the research in this area assumes, or is restricted to, traditional software development approaches. By excluding newer ways of developing software, such as agile software development [1], important aspects of KM in software engineering have been omitted.

The suggestions made by for example Backlund [2] and Wastell [22] perfectly agree with the principles of the Agile Manifesto, which clearly states the value of individuals and interactions and constant customer collaboration for successful software development project outcomes [1]. The agile principles of daily collaboration between business people and developers, of face-to-face conversation for conveying information and knowledge, of building projects around motivated individuals and of continuous reflection to adjust behavior are only a few examples demonstrating what needs to be in place for a software project to succeed. This makes the agile family of methods a promising candidate to investigate from a KM perspective. Moreover, by elucidating some of the general benefits and challenges of KM and by making them explicit to the agile community may provide insights that could further enhance agile methods, such as eXtreme Programming (XP) [4] [5].

In this paper, the result of a theoretical analysis of aspects of knowledge transfer in XP is presented as a first attempt to clarify and elucidate the relationship between these two domains. The analysis is based on XP as a representative of the group of agile software development methods because XP is the agile method that is most widely used [9][1].

2. WHY KM AND XP?

In today's competitive economy where many organisations uncover the most opportunities from intellectual rather than physical capital [14], it is crucial that knowledge is managed. Knowledge management in software teams is no exception. The large amount of knowledge acquired during software development, e.g. knowledge associated with the development process, the business domain of the project and the developed software [2][13] must be managed for several reasons: (a) as a means to help software development teams to be able to leverage that knowledge and (b) which can prevent software projects from failing as discussed previously [2] [21] [22]. The fact that, organisations that make sure that knowledge transfer is taking place in a systematic and controlled manner have shown to demonstrate great cost savings [6] [12] [14] [16], should be enough as argument in this context. More importantly, the argument applies to XP teams as well. XP is a minimalist, highly disciplined software development method, which when used appropriately, has a proven capability to produce planned results in a timely manner. However, potential method improvements can and should always apply. To be even more efficient, effective, productive and successful, XP teams need to be equipped with appropriate support to further improve and facilitate the development of software. This could, in certain contexts, be achieved by introducing and by clarifying the value of KM to XP. For example, in terms of supporting scalability of XP which is an issue that has been heavily debated over the past years, e.g. by Boehm and Turner [6], Cao et al. [7], Crocker [8], Paulk [17] and Reifer et al. [18]. In small co-located XP teams, managing knowledge and knowledge transfer may not be an issue, but for larger or distributed XP teams, it is believed that KM may provide valuable support for scaling XP. Effective management of knowledge between projects is another issue that calls for further exploration. For example, finding patterns or techniques, e.g. on the basis of retrospectives [21], for fine-tuning behaviour also in inter-project situations could help in achieving gains in performance. Thus, it is argued that a structured analysis of XP from a KM perspective is needed. It is important to gain understanding of how the two fields relate as well as to investigate to what extent aspects of KM can be beneficial to XP. In the following, the results of the first step towards this goal will be discussed.

3. CHALLENGES OF KNOWLEDGE TRANSFER

Knowledge is transferred in organisations everyday, in a controlled or/or uncontrolled way. Ideas exchanged by people in the hallways, information forwarded in emails or posted on notice boards are only a few examples of everyday knowledge transfer. The question is how an organisation or a team can transfer knowledge effectively so that the whole organisation or team can truly benefit from it.

There are several theories about how to control knowledge transfer. For example, Dixon [12] presents a spiral model specifying eight steps that must be taken for creating and leveraging knowledge within an organisation. The model specifies three criteria that determine how a particular transfer method will work in a particular situation. The three criteria: who the intended receiver is, nature of task and type of knowledge to be transferred have an impact on the choice transfer method and how knowledge can be translated into a usable form [12]. Davenport's and Prusak's [11] reasoning about successful knowledge transfer involves the same ideas. When knowledge is to be transferred, the transfer method must always suit the culture and this involves considering factors that have an impact on the transferring process, such as the kind of knowledge to be transferred and the culture of the team [11]. The latter have also identified a set of common cultural factors, "frictions", which challenge effective knowledge transfer. In Table 1, the frictions and possible solutions to prevent these frictions from occurring are summarised [11].

4. ANALYSIS OF XP FROM A KM PERSPECTIVE

KM is a large multi-disciplinary field. Therefore, it necessarily covers a larger domain than relevant for the analysis focused on in this paper. For example, it specifies a number of areas that primarily describe KM practices and activities from a business and managerial point of view, which are not always specific to software engineering or XP [11][12][14][16]. However, this research aims at clarifying and elucidating the relationship between KM and XP, starting with an investigation to determine whether, and to what extent, XP deals with some of the major challenges of effective knowledge transfer as presented in Table 1 [11]. Consequently, the analysis focuses on this topic.

XP is analysed as a whole method, including its values, principles and primary practices [5]. However, individual parts of XP are discussed when specifically addressing any of the cultural frictions. In Table 2, a summary of the results of the analysis is presented. In the second column, the XP values, principles and practices that are comparable to the solutions suggested by Davenport and Prusak [11] are indicated.

According to Davenport and Prusak [11], the values, norms and behaviours that make up a team's culture are the principal determinants of how successfully important knowledge is created and transferred. This analysis shows that all the XP values, principles and primary practices are in various degree counterparts to the solutions to effective knowledge transfer [11]. In the following subsections an accompanying motivation for the results in Table 2 is presented.

4.1 Values

All of the frictions that can inhibit effective knowledge transfer are accounted for in the values of XP. Lack of trust, lack of time and meeting points, having people from different cultures on the team, intolerance for mistakes and beliefs that knowledge is ascribed to certain individuals or groups [11] are covered by the values of communication, respect, feedback, courage and simplicity [4] [5].

Table 1. Cultural factors that inhibit knowledge transfer and possible solutions [11]

FRICTION	POSSIBLE SOLUTIONS
Lack of trust	Build relationships and trust through face-to-face meetings
Different cultures, frames of reference, vocabularies	Create common ground through education, discussion, teaming, publications, job rotation
Lack of time and meeting places	Establish times and places for knowledge transfers: talk rooms, fairs, conference reports
Status and rewards go to knowledge owners	Evaluate performance and provide incentives based on sharing
Lack of absorptive capacity in recipients	Educate employees for flexibility; provide time for learning; hire for openness to ideas
Belief that knowledge is prerogative of particular groups, "not invented here-syndrome"	Encourage non-hierarchical approach to knowledge; quality of ideas more important than status of source
Intolerance for mistakes or need for help	Accept and reward creative errors and collaboration; no loss of status from not knowing everything

Table 2. How XP deals with "frictions", i.e. with barriers to effective knowledge transfer

FRICTION	XP		
	Values	Principles	Practices
Lack of trust	Communication; Respect; Courage; Feed-back	Humanity; Mutual benefit; Accepted responsibility; Improvement; Reflection; Opportunity	Whole team; Sit together; Informative workspace; Pair programming; Slack
Different cultures, frames of reference, vocabularies	Communication; Respect; Courage; Feed-back; Simplicity	Humanity; Diversity; Mutual benefit; Improvement; Reflection; Opportunity	Whole team; Sit together; Informative workspace; Pair programming; Stories; Test-first programming
Lack of time and meeting places	Communication; Respect; Feed-back; Simplicity; Courage	Mutual benefit; Diversity; Improvement; Reflection; Opportunity	Whole team; Sit together; Informative workspace; Pair programming; Weekly/Quarterly cycles
Status and rewards go to knowledge owners	Respect; Feed-back; Communication; Courage	Humanity; Failure; Mutual benefit; Reflection; Opportunity	Whole team; Sit together; Informative workspace; Pair programming; Weekly/Quarterly cycles; Continuous integration
Lack of absorptive capacity in recipients	Communication; Respect; Feed-back; Courage; Simplicity	Humanity; Failure; Mutual benefit; Reflection; Opportunity	Whole team; Sit together; Informative workspace; Pair programming
Belief that knowledge is prerogative of particular groups, "not invented here-syndrome"	Respect; Communication; Feed-back; Courage; Simplicity	Humanity; Failure; Mutual benefit; Reflection; Opportunity; Improvement	Whole team; Sit together; Informative workspace; Pair programming
Intolerance for mistakes or need for help	Respect; Communication; Feed-back; Courage; Simplicity	Humanity; Failure; Mutual benefit; Reflection; Opportunity; Improvement	Whole team; Sit together; Informative workspace; Pair programming; Test-first programming

Communication in XP means building relations by conveying information to and within a development team through openness and steady face-to-face conversation, which prevents frictions such as lack of trust and lack of understanding from occurring.

Feedback is closely related to communication and addresses the value of keeping the project and the team on track. It is also emphasised that feedback is most useful if it is done rapidly. The time between an action and its feedback is critical to learning and the ability to making any changes or corrections needed [4]. In this respect, XP encourages effective knowledge transfer, knowledge creation and the establishment of learning environments. More importantly, the overall goal of communication and feedback is to achieve a shared understanding among team members to strengthen teambuilding and the feeling of shared project responsibility, which prevents status and rewards from going to knowledge owners but which also discourages a hierarchical approach to knowledge.

To accomplish good communication and feedback requires respect and tolerance among team members. Therefore, respect is one of the core values in XP. By respecting other peoples' backgrounds and frames of reference, common grounds for constructive collaboration, discussion and knowledge and information exchange can be built. In this context, respect also mitigates the risk for intolerance for mistakes. XP instead pushes for creativity and time for learning. Errors are regarded as a mechanism to learning rather than failure. To be creative and to learn new things also requires courage, which is emphasised through the value of courage in XP.

Related to all the previous values is simplicity, which facilitates communication, feedback as well as courage. For example, a simple design with very simple code or simple documentation can be easily understood by every person in the team, whether a customer or programmer, which also provides a foundation for effective knowledge transfer.

4.2 Principles

The XP principles can be regarded as counterparts to the solutions to overcome cultural frictions in knowledge transfer presented by Davenport and Prusak [11]. The XP principles of humanity, mutual benefit, diversity, failure, reflection, opportunity and improvement are particularly comparable to common ways of resolving cultural problems of knowledge transfer.

The principle of humanity signifies the need to meet basic human needs in projects, such as building relationships and connection, which is primarily accomplished by communication in XP. The most effective way to communicate is through face-to-face conversation because conversation allows dialogue, i.e. it allows explanation, clarification and direct feedback. In this respect, humanity facilitates building relationships which creates incentives to share ideas and knowledge and therefore amplifies knowledge transfer. Davenport and Prusak [11] state that conversation is fundamental to knowledge transfer because it provides a simple way to discover what people know and to share knowledge with colleagues, which results in new or improved knowledge for the whole organisation.

The principle of mutual benefit is probably the most important of all the XP principles. Basically, it means maintaining good working relationships in a project team. By making sure that everyone on a project team are benefited from any project activities, e.g. documentation, provides a starting place for good working environments and gives a concrete incentive for people to share and exchange ideas.

The principle of diversity suggests that effectiveness comes from teams where people come from different backgrounds, with different skills and perspectives. The key is, at any stage of the project, to make people with important or necessary skills available as a resource for a project to succeed. This principle therefore undoubtedly encourages knowledge transfer.

Failure is another principle that clearly encourages knowledge transfer. More specifically, this principle addresses the problem of intolerance for mistakes or need for help which can inhibit effective knowledge transfer. Because valuable knowledge can sometimes be difficult to obtain, one way is to try out different solutions where failure can provide an important learning mechanism [5].

Related to failure are the principles of opportunity, reflection and improvement. In XP, improvement is imperative. To reach quality and excellence, problems need to be seen as opportunities for change, for learning and for improvement. For this to take place, a team needs to reflect on their work, i.e. on how they work. This is realised through project retrospectives [19] or similar techniques, which include analysing factors of success and failure and to openly expose them and to learn from them.

4.3 Practices

The practices of XP can be viewed as "techniques for rapidly building and disseminating knowledge among members of a development team" [5], where the goal is to have all team members share the same view and expectations of the system and the project. As indicated in Table 2, this is primarily achieved by the XP practices of whole team, sitting together, informative workspace, stories, incremental design and pair programming. The first three practices are tightly interrelated. The basic idea of whole team in XP is that the team consists of people with the all skills and knowledge necessary for a project to succeed, which includes both customers and developers. The aim is also to build projects around motivated individuals who share the responsibility for supporting each other's work and the growth and learning of the whole team [5].

The practice of whole team is reinforced by the practice of sitting together. In XP, the whole team works co-located in an open space, which provides time for physical proximity and for meeting and discussing face-to-face. It is argued that the more time the team works physically co-located, the more humane and productive the project [5]. By working together and physically closer to each other improves communication, makes it easier to build relationships and naturally establishes time and places for knowledge transfers, which are keys to overcome many of the cultural barriers to effective knowledge transfer.

In addition, the practice of informative workspace, e.g. a big visible chart, does not only convey project related information to any interested stakeholder, it also serves as a natural meeting point at any time because this is where an XP team gathers for the daily stand-up meeting [4], for project planning or for any discussion for that matter. This prevents a situation where difficulties with finding meeting places would occur. Furthermore, the fact that informative workspaces are visible and open to everyone provides a mechanism for spreading information and knowledge itself, which is also reflected by the name of the practice. In this way, also the problem of beliefs that information or knowledge is sanctioned to particular people, i.e. the "not invented here-syndrome" [11], is solved.

Stories create common ground for discussion and exchanging ideas and knowledge about the system to be developed. Stories also help the team to find common vocabularies and to define a common frame of reference which facilitates knowledge transfer.

From a knowledge transfer perspective, the XP practices of incremental design, weekly and quarterly cycle allow a team to collect information, learn the system and the technology and understand customer needs in small, iterative steps. Such an approach provides time for feedback and for making timely changes and refinements rather than making early design decisions based on speculative guesses which can have a negative impact on the learning process as well as on the project results.

Pair programming involves many well-known knowledge transfer techniques, such as pair work, shadowing and mentoring [12]. More specifically, pair programming provides space for dialogue between two persons and aims to increase overall quality by offering team members the possibility to clarify ideas, exchange knowledge and help each other with the tasks at hand. This is regarded best practice for transferring or recreating tacit knowledge [14]. With shadowing for example, a less experienced developer observes more experienced developers in their activities to learn how their more experienced counterparts approach their work. By discussing his or her observations with the expert, tacit knowledge is made explicit and knowledge transfer is truly achieved [14]. In addition, as all pairs in an XP team rotate on a regular basis enables spreading knowledge about the project throughout the entire team which mitigates the risk of loosing critical knowledge if a person would leave the team.

5. CONCLUSIONS

To successfully transfer an individual's knowledge in practice, a team must actively work out ways to make personal knowledge available to others. This is a central activity in the knowledge creating company [16]. By providing guidance on how to build relationships, create incentives for sharing knowledge, build up acceptance for creative errors and for learning in software projects, this analysis shows that good support for knowledge creation and knowledge transfer can be identified in all the values, principles and practices of XP. More specifically, based on this analysis, our conclusion is that within a single team XP practices are sufficient for KM and no separate KM is needed. Based on our experiences, the real KM challenge lies in how to share the knowledge created in XP teams with other parts of the organisation. By using the framework used in this analysis it would be possible to analyse how XP supports KM in multi-team settings. Our understanding is that XP does not support KM in these cases. More specifically, the result gives substantial support to an underlying hypothesis that KM can provide valuable support to large XP teams, thereby facilitating scalability - an issue that has been heavily debated over the past years. If the hypothesis were verified, the KM effects measured and empirically validated would provide a good foundation for designing recommendations for how to use KM when scaling up XP.

REFERENCES

[1] Agile Manifesto, Agile Manifesto. URL: http://www.agilemanifesto.org. Accessed in June 2005.

[2] Backlund P., Development Process Knowledge Transfer through Method Adaptation, Implementation and Use. Institutionen för Data och Systemvetenskap, Stockholms Universitet/KTH, Sweden, Ph D Thesis, 2004.

[3] Basili V., Caldiera, G., and Rombach D., The Experience Factory. Encyclopedia of Software Engineering - 2 Volume Set, pp.469-476, 1994.

[4] Beck K., Extreme Programming Explained: Embrace Change. Addison-Wesley, 2000.

[5] Beck K., Extreme Programming Explained: Embrace Change. 2nd Edition. Addison-Wesley, 2004.

[6] Burke G. and Howard W., Knowledge Management and Process Improvement: A Union of Two Disciplines. URL: http://www.stsc.hill.af.mil/crosstalk/2005/06/0506Burke.html. Accessed in December 2005.

[7] Cao L. et al., How Extreme does Extreme Programming have to be? Adapting XP Practices to Large-scale Projects. Proceedings of the 37th Hawaii International Conference on System Sciences, 2004.

[8] Crocker R., The Five Reasons XP Can't Scale and What to Do About Them. URL: www.agilealliance.org/articles/ articles/FiveReasonsXPCantScale.pdf. Accessed in June 2003.

[9] Charette R., The Decision is in: Agile versus Heavy Methodologies. Agile development and Project Management, Cutter Consortium, Vol. 2 (19), URL: www.cutter.com/freestuff/epmu0119.html. Accessed in February 2004.

[10] Cutter Consortium, URL: http://www.cutter.com/research/2004/edge040427.html. Accessed in June 2005.

[11] Davenport T. and Prusak L., Working Knowledge: How Organisations Manage What They Know. Harvard Business School Press, 1998.

[12] Dixon N., Common Knowledge: How Organisations Thrive by Sharing What They Know. Harvard Business School Press, 2000.

[13] Iivari J., Informations Systems Development as Knowledge Work: The Body of Systems Development Process Knowledge. In Information Systems Modelling and Knowledge Bases XI. (Eds. Kawaguci E. et al.), IOS Press, 2000.

[14] Levinson M., The ABCs of Knowledge Management. CIO Magazine, URL: http://www.cio.com/research/knowledge/edit/kmabcs.html#support. Accessed in December 2005.

[15] Marwick Alan D., Knowledge Management Technologies. IBM Systems Journal, Knowledge Management, Vol. 40 (4), 2001.

[16] Nonaka I., The Knowledge Creating Company. Originally published in December 1991. Reprint 91608, Harvard Business Review on Knowledge Management Paper Back Series. Harvard Business School Press, 1998.

[17] Paulk M., Extreme Programming from a CMM Perspective. IEEE Software, Vol. 18 (6), 2001.

[18] Reifer D.J., Maurer F. and Erdogmus H, Scaling Agile Methods. IEEE Software, Vol. 20 (4) pp.12-14, 2003.

[19] Rising L., Patterns and Retrospectives. *Cutter IT Journal*, Vol. 16 (9), 2003. URL: http://members.cox.net/risingl1/articles/Cutter.pdf. Accessed in July 2005.

[20] Standish Group, CHAOS Report. URL: http://www.standishgroup.com/sample_research/chaos_1994_1.php. Accessed in June 2005.

[21] Ye Y. and Fay R., Knowledge Transfer through Asymmetric Pair Programming. URL: http://www.agilealliance.org/articles/articles/AsymmetricPairProgramming.pdf. Accessed in June 2005.

[22] Wastell D. G., Learning Dysfunctions in Information Systems Development: Overcoming the Social Defenses with Transitional Objects. MIS Quarterly, Vol. 34 (4), 1999.

ENDNOTE

1 XP and its values, principles and practices will not be presented in detail, but we refer to Beck [4][5] when any specific XP concepts are used in the paper.