Managing Knowledge in Agile Software Development

Amitoj Singh
Assistant Professor
Patel Memorial National College, Rajpura

Kawaljeet Singh
Director,
University Computer Centre,
Punjabi University, Patiala

Neeraj Sharma
Associate Professor
Punjabi University, Patiala

ABSTRACT

These days business activities are changing at very rapid rate and there are increasingly complex requirements set on programming solution that puts traditional software development methods (also called heavyweight) at the rear and leads to the need for other development practices which can overcome the problem of software crises. Modern approaches, also known as agile or lightweight methodologies, claim to provide solution to above said problem. Heavyweight methodologies, commonly known for its traditional ways to develop software put emphasis on comprehensive planning, detailed documentation, and expansive design. Unlike traditional methods, agile methodologies employ short iterative cycles, and rely on tacit knowledge within a team. Knowledge Management (KM) can be easily accepted into agile software development environments. Following are two reasons in favor of this point of view. First, the agile cultural infrastructure already encourages values such as cooperation, communication and knowledge sharing; specifically, agile software development processes include some practices that support KM, e.g. stand-up meetings, the planning game, pair programming and the informative workplace. Second, KM is about learning, and ASD set up an environment that supports learning processes. In this paper, attempt is made to find out specific agile practices which promote KM.

General Terms

Knowledge Management, Agile software development

Keywords

Scrum, Extreme programming, Knowledge Sharing

1. INTRODUCTION

For many years a large number of different software development methodologies have been proposed to tackle the problems associated with software development. Avison and Fitzgerald define methodology as a set of procedures, techniques, tools and documentation aid which help developers to implement a new information system [1], [37]. Methodologies help in imposing a closely controlled process upon software development with a goal to make a software development process more efficient and predictable [2].

One of the reasons for using these methodologies is to put in order the different work processes by emphasis on planning, and so that it can help the development team to succeed in the project in which they are working. Another reason is to make software development process more predictable and efficient [3]. Traditional methodologies are plan-driven i.e. work begins with the elicitation and documentation of an entire set of requirements, complete architectural plan and followed by high-level design. Traditional methodologies emphasize on complete planning, detailed documentation, and expansive design. Because of these heavy features, this methodology is popularly known as heavyweight. But in spite of these features, lots of projects get failed, over budget or delayed. Lot of studies have been conducted which have reported the failures with these heavyweight methodologies. These failures are also referred to as 'software crises (e.g. [4],[5]).

Glass (2001) makes use of the term 'methodology wars' to illustrate the often unfriendly dispute between advocates of agile and plan-driven software development. He further added that with the publication of the 'Manifesto for Agile Software Development' (Agile Alliance 2001) [7] the dispute became more severe [6]. Advocates of agile argued that these methodologies cope successfully with common problems of software projects. To support this argument many surveys have been carried out (e.g. [34], [35], [36]).

2. AGILE SOFTWARE DEVELOPMENT

What is the meaning of being agile? According to Jim Highsmith being agile means being able to deliver quickly, change quickly, and change often [8]. With an agile approach, one can deliver business-oriented results rapidly and effectively. Differentiating from the working point of view of traditional methods, agile methodologies use short iterative cycles, and rely on tacit knowledge within a team. The name lightweight or agile can be defined as "1) marked by ready ability to move with quick easy grace or 2) having a quick resourceful and adaptable character" [9]. An agile method generally encourages incremental development and delivery of software product. This process should be able to allow changes occurring during the development phase and should be adaptive in nature [10].

The name "agile" came about in 2001, when seventeen process methodologists held a meeting to talk about the future trends in software development. The outcome to this meeting was the formation of "Agile Alliance" and its manifesto for agile software development.

2.1 Features of agile manifesto [7]

- Individuals and interactions over processes and tools.
- Working software over comprehensive documentation.
- Customer collaboration over contract negotiation.
- Responding to change over following a plan.

Agile techniques may differ in practices (XP, ASD, DSDM, SCRUM, CRYSTAL etc.) but, they contribute to common characteristics, including iterative development, and a focus on interaction and communication. Cockburn and Highsmith explain what is new about agile methods is not the practices they use, but their recognition of people as the main driving force which can lead to project success [11]. Many surveys have claimed XP and Scrum methodologies are widely used in the industry and are the most popular (e.g. [34], [35], [36]). These methodologies are explained in brief.

2.2 Extreme Programming (XP)

Extreme Programming (XP) is the agile methodology that has gained most attention in the last few years. Extreme Programming (XP) was developed before the Agile Manifesto was written and Principles of agile manifesto have influenced this methodology. Extreme Programming (XP), created by Kent Beck, have four fundamental values: communication, feedback, courage, simplicity. A summary of XP practices are given below [38].

2.2.1 Planning

Efforts needed to implement customer stories have been estimated by programmer and the customer decides the scope and timing of releases based on estimates.

2.2.2 Small/short releases

Series of small and regularly updated versions of application is developed. New versions are released anywhere from daily to monthly.

2.2.3 Refactoring

Refactoring refers to restructuring the system. It involves adding flexibility and simplification of the code without changing its functionality and removing duplication of code if it exists.

2.2.4 Pair programming

Two Programmers work on a single computer. One person types the code and is called the driver, another reviews it and is called the navigator. Role of the navigator is to check the code for errors simultaneously.

2.2.5 Test first development

In this approach, the automated tests are written prior to the writing of functional code. It is similar to Test Driven Development (TDD).

2.2.6 Collective ownership

Collective Code ownership means no single person possess or is responsible for individual code segments, rather anyone can change any part of the code at any time.

2.2.7 Metaphor

It is defined by a set of metaphors between the customer and the programmers which describes how the system works.

2.2.8 Continuous Integration

Continuous Integration means a new piece of code is integrated with the existing system when it is ready to use and while integrating new code to system, the system is built again and all tests are performed on integrated system for the changes to be accepted.

2.2.9 On-site customer

It means all the time customer will be available to the development team.

2.3 SCRUM

The term 'scrum' was originally derived from the game of rugby where it means "getting an out-of-play ball back into game" with teamwork. Scrum does not provide any specific practices for software development but it provides a management strategy or tools to control the development process and to avoid the chaos by unpredictability and complexity. In Scrum, software is delivered in increments called "Sprints". Each sprint begins with planning and ends with a review. A sprint planning which is done by Scrum team is a time-boxed event, which is used for detailed planning for sprint. The stakeholders of a project attend sprint review meetings to review the state of the business, market and technology. A retrospective meeting is used to assess the degree of teamwork in the completed sprints. Some of the key Scrum practices are given below [39].

2.3.1 Product Backlog

This is the prioritized list of all features and changes that are yet to be made to the system. The product Owner is responsible for maintaining the Product Backlog.

2.3.2 Sprints

Sprints are 30-days in length. These are the iterations in which all the development work is done.

2.3.3 Sprint Planning Meeting

Sprint planning meeting is attended by the customers, users, management, Product owner and Scrum Team. This meeting is used to set goals and functionality of the product.

2.3.4 Daily Scrum Meeting

It is a daily meeting for approximately 15 minutes, which is organized to keep track of the progress of the Scrum Team and address any problem faced by the team

3. KNOWLEDGE MANAGEMENT

Software development process has always been a knowledge-intensive task. As the complexity of software building has increased, there is a greater need of knowledge processes to solve the problems [12], [13]. Knowledge management is "a method that simplifies the process of sharing, distributing, creating, capturing and understanding the company knowledge" [14]. Knowledge is one of the main competitive assets of the organization, which allows the enterprise to be productive and to deliver competitive products and services. Companies and organizations can improve their ability to create, acquire, disseminate, and retain knowledge simply by applying knowledge management techniques, thus allowing them to make efficient decisions, control complexity, and improve productivity [15].

Nonaka [16] differentiates 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. Classifying it further, Spender [17] categorizes knowledge as implicit, explicit, individual and collective knowledge. It is common belief that implicit and explicit is important however, implicit knowledge is more difficult to identify and manage [18].

But in practice, organisations deal with two fundamental and opposing knowledge strategies, these are codification strategy or a personalization strategy [19].

3.1.1 Codification

To arrange and store information that constitutes the knowledge of the company, and to make this knowledge available to the people working in the organisation.

3.1.2 Personalization

It maintains the flow of information in a company by having a centralized store of information about knowledge sources, like a "yellow pages" of who knows what in a company.

Earl [38] has classified work in knowledge management into schools. The schools are broadly categorized as “technocratic”, “economic” and “behavioural”. The technocratic schools are 1) the systems school, which focuses on technology for knowledge sharing, using knowledge repositories; 2) the cartographic school, which focuses on knowledge maps and creating knowledge directories; and 3) the engineering school, which focuses on processes and knowledge flows in organizations.

4. AGILE KNOWLEDGE MANAGEMENT

Knowledge Management (KM) has emerged in response to the importance of knowledge and the need to maximize its usefulness. Much of early KM research was focused on Information Technology (IT), often used for building software systems to explicitly record, organize, and disseminate knowledge within an enterprise. The range of KM research has since widen and become more multidisciplinary. Specifically, KM researchers are emphasizing on the social and tacit aspects of knowledge which affect the software development process especially in this era where software development is done by distributed teams of different cultural and social background. The challenge for future KM strategies is to address these social and tacit aspects of knowledge [20], [21], [22].

KM and Agile Software Development (ASD) are two organizational processes that face common barriers when introduced and applied in software development. The main barrier in initiating the product development in agile software development and implementing knowledge management into software organizations is the need to deal with the conceptual change, mainly the organizational cultural change that ASD and KM brings when introduced. Many studies have revealed that the introduction of KM and ASD processes have increased productivity, shortened time-to-market and resulted in higher product quality (e.g. [29], [30]). The major 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 priceless knowledge relating to product development, the software development process, project management and technologies which go along with the developer.

Boehm and Turner [23] note that agile methods rely on tacit knowledge and it depends on the ability to cultivate and share it. That is why management of workers’ knowledge is major concern in agile methods. The pairing of KM and ASD is not new; a connection between the two concepts has been recognized by various researchers [31], [32]. This association, however, is not surprising because both disciplines deal with organizational culture and change management.

KM has the potential to be easily accepted into ASD environments. Following are two clarifications for this viewpoint. First, the agile cultural infrastructure already includes values such as cooperation and knowledge sharing; specifically, ASD processes include some practices that support KM, such as stand-up meetings, the planning game, pair programming and the informative workplace. Second, KM is about learning, and ASD establishes an environment that supports learning processes [28]. There are many agile practices that foster KM in agile projects

4.1 On Site Customer

In agile software development methods, the product customer is a part of the development process. This direct communication channel increases the probability that the software requirements are communicated correctly and it helps to deal successfully with change introduction at later stages. On site customer helps in developing new knowledge and using that knowledge by directly communicating with software development teams and producing regular feed for moving the project according to ones need.

4.2 Pair Programming

Pair programming is a practice of XP that contributes to an effective knowledge management framework. In this practice two developers work together on a single computer collaborating on the same analysis, design implementation, and testing. Pair formation and pair rotation also serves the purpose of preserving and sharing knowledge among the team members. Programmers learn and become more skillful by working in coordination. In case the developer leaves the project, there need not be a loss of momentum or impact to project schedules because there is always at least one person that can provide the relevant knowledge.

4.3 Collective Code Ownership

Collective code ownership means that everyone is responsible for all of the code and its not single person’s property. This means that everyone must have confidence in every member of the team. But it also means that there must be a provision for everyone to acquire everyone else’s specialized knowledge. This helps in developing new knowledge. No one owns code. Any developer is expected to be able to work on any part of the code at any time. This helps in sharing and preserving one’s knowledge with another.

4.4 Collaborative Workspace

The walls of the development workspace (either virtual or physical) serve as a communication means. 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, this helps in dissemination and sharing of knowledge as all project stakeholders can be updated at a glance at any time about the project progress and status.

4.5 Whole team

The practice of Whole Team also promotes KM as the development teams (which include all kinds of roles) communicate face-to-face. The whole team practice can be implemented in several ways, e.g. all the different roles in traditional teams are merged together to make a team, these are co-located teams and use space to maximize the communication and knowledge sharing among team members which hold different roles. All team members participate in team meetings to review the planning process and get feedback from the customer about requirements [24], [25].

4.6 Stand-up meetings

The entire team comes together for a daily stand-up meeting which is organized to keep track of the progress of the Scrum Team. It helps in learning process as people share their successes and failure in these meetings, each team member presents the status of his or her development tasks and what he or she plans to accomplish during the days to come, both with respect to the development tasks and the personal role.

Better communication helps in sharing personal views by each team with respect to anticipated problems.

4.7 Sprint planning meeting

Sprint planning meeting is always conducted before the starting of new sprint. The meeting is attended by the customers, users, management, product owner (onsite customer) and Scrum team where the goals state and functionality of the code which will be the outcome of the sprint is decided. These meetings help in understanding the actual requirements of the customers and the structure which is used to implement these requirements. These kinds of planning meeting help in transferring and preserving knowledge among team members.

4.8 Roles

Within the whole team concept, each team has an additional role of acting as a team leader, because it does not matter how skilled you are, it is impossible to handle all the essential and complex responsibility of software project. The allocation of responsibility through roles helps in better management of the project. This means all team members are involved in all parts of the developed software [24], [25].

5. Conclusion

In today’s competitive and complex global market, companies are required to manage their intellectual resources as well as their financial resources. Therefore, KM is accepted as a genuine management practice that helps organizations to distribute the right knowledge to the right people at the right time [33]. The ASD approach emerged over the past decade in response to the unique problems that characterize software development processes. In general, ASD emphasizes customer needs, communication among team members, short releases and heavy testing all through the entire development process. These ideas are implemented by the different agile development methods. Agile methodologies promote knowledge dissemination, retention, and often informal sharing of tacit knowledge among the team members. However agile practices do not support acquisition of knowledge about new technologies very much. Because the priority is quick delivery of working system, the time which can be spent in acquiring knowledge about new technologies is greatly reduced. On the other hand, acquiring of domain knowledge is highly promoted by agile, because its processes prescribe a tight cooperation with customers and their representatives.

6. REFERENCES

[1] Beck, M. K. 1999 Embracing change with Extreme Programming. IEEE Computer, Vol. 32.

[2] Awad, M. A. 2005 Comparison between Agile and Traditional Software Development Methodologies, Honours Programme Thesis, University of Western Australia.

[3] Fowler, M. 2000 The New Methodology , http://www.martinfowler.com/articles/newMethodology.html accessed on 11.11.11

[4] Lycett, M. Macredie, R. Patel, C. and Paulk, R. J. 2003 Migrating Agile Methods to Standardized Development Practice, Computer, Vol. 36, issue 6, pp. 79-85.

[5] The CHAOS Report, 1994 www.standishgroup.com/sample_research/chaos_1994_1.php, accessed on 15.10.11

[6] Glass, R. L. 2001 Agile Versus Traditional: Make Love, Not War, Cutter IT Journal, Vol. 14, No. 12, pp12-18.

[7] Agile Alliance 2001 Manifesto for Agile Software Development, www.AgileAlliance.org accessed on 12.11.11

[8] Highsmith, J. Orr, K. and Cockburn, A. 2000 Extreme Programming, E- Business Application Delivery, pp. 4-17

[9] Job Trends www.indeed.com/jobtrends?q=agile%2C+scrum%2C+%22extreme+programming%22%2C+%22test+driven%22&l=,accessed on 11.11.11

[10] Tudor, D.2006 An update on Agile methods, ITadviser, Issue 56.

[11] Highsmith, J and Cockburn, A. 2001 Agile Software Development: The Business of Innovation, Computer, Vol. 34,No. 9, pp. 120-122.

[12] Desouza, K.C. 2003 Barriers to effective use of knowledge management systems in software engineering, Communications of the ACM, Vol; 46 (1). pp 99–101.

[13] Disterer, G. 2002 Management of project knowledge and experiences, Journal of Knowledge Management Vol. 6 (5) pp. 512–520.

[14] Davenprot, T. H. Prusak, L.1998 Working Knowledge: how organizations Manage what they Know, Harvard Business School Press, Boston, USA.

[15] Tiwana, A, 2000 The Knowledge Management Toolkit, Upper Saddle River, NJ: Prentice Hall PTR, Prentice-Hall, Inc.

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

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

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

[19] Hansen, M.T., Nohria, N. and Tierney, T. 1999 What's Your Strategy for Managing Knowledge?, Harvard Business Review, March/April, pp106-116 Jashapara, A. (2004) Knowledge Management, Prentice Hall.

[20] Abou-Zeid, E. S. 2002 A knowledge management reference model. Journal of Knowledge Management, Vol. 6(2) pp. 486-499.

[21] Manville, B. 1999 Complex adaptive knowledge management, in The Biology of Business, J.C. III, Editor. 1999, Jossey-Bass: San Francisco. pp 89-111

[22] Bonifacio, M. Bouquet, P. and Traverso, P.2002 Enabling Distributed Knowledge Management: Managerial and Technological Implications. Novatica and Informatik/Informatique, Vol. 3(1) pp, 22-29

[23] Boehm, B. and Turner, R. 2004 Balancing Agility and Discipline, Addison-Wesley.

[24] Hazzen, O. Levy, M. 2009 Knowledge Management in Practice:The Case of Agile Software Development. In Proceeding of ICSE Workshop on Cooperative and Human Aspects on Software Engineering.

[25] Hazzan, O. and Dubinks . 2008 Agile Software Engineering, Undergraduate Topics in Computer Science (UTiCS) Series, Springer.

[26] Ding, W. and Marchionini, G. 1997 A Study on Video Browsing Strategies. Technical Report. University of Maryland at College Park.

[27] Fröhlich, B. and Plate, J. 2000. The cubic mouse: a new device for three-dimensional input. In Proceedings of the SIGCHI Conference on Human Factors in Computing Systems

[28] Hazzan, O. and Dubinsky, Y. 2003. Teaching a Software Development Methodology: The Case of Extreme Programming, In proceedings of the 16th International Conference on Software Engineering Education and Training, Madrid, Spain, pp. 176-184.

[29] Bennet, D. and Bennet, A. 2003 The Rise of the Knowledge Organisation, chapter 1 in Holsapple, C.W. (ed.), Handbook on Knowledge Management,Vol 1, Springer, Berlin, pp. 5–20.

[30] Reifer, D. J. 2002 How to Get the Most out of Extreme Programming/Agile Methods , In proceeding Proc. Extreme Programming and Agile Methods - XP/Agile Universe

[31] Dove, R. 1999 Knowledge management, response ability, and the agileenterprise", Journal of Knowledge Management, Vol. 3 Iss: 1, pp.18 - 35

[32] Harald Holz, Melnik, G. Schaaf, M.: Knowledge Management for Distributed Agile Processes: Models, Techniques, and Infrastructure. WETICE 2003 pp 291-294

[33] Spek, R. Kruizinga, E Annelies Kleijsen, 2009 Strengthening lateral relations in organisations through knowledge management, Journal of Knowledge Management, Vol. 13 Iss: 3, pp.3 – 12

[34] Version one, 2008 3rd Annual Survey: The State of Agile Development, www.versionone.com/pdf/3rdAnnualStateOfAgile_FullDataReport.pdf.

[35] Version one, 2007 Agile development: Result Delivered, http://www.versionone.net/pdf/AgileDevelopment_ResultsDelivered.pdf.

[36] Versiion One, 2009 4th Annual Survey 2009, www.versionone.com/agilesurvey.

[37] Avison, D.E. and Fitzgerald, G. 1995. Information Systems Development: Methodologies, Techniques and Tools. 2nd ed., Maidenhead: McGraw Hill

[38] Beck, K. 1999 Extreme programming explained: embrace change, Addison-Wesley Longman Publishing Co., Inc. Boston, MA, USA

[39] Schwaber, K. and Sutherland, J. 2011 The Scrum Guide, www.scrum.org/storage/scrumguides/Scrum%20Guide%20-%202011.pdf