Information and Knowledge Management
ISSN 2224-5758 (Paper) ISSN 2224-896X (Online)
Vol.3, No.3, 2013

Knowledge Management: the agile way

Amitoj Singh[1*] Kawaljeet Singh[2] Neeraj Sharma[2]

1. PMN College, Rajpura, Punjab, India
2. Punjabi University, Patiala, Punjab, Inida

* E-mail of the corresponding author: amitoj@mail.com

Abstract

The logic behind the acceptance of agile practice is that along with software development, project and knowledge management (KM) practices are woven into the practices of agile methodologies, which have made these methodologies very popular among software development communities. There are many practices in agile (e.g. Pair programming, scrum meetings, onsite customer etc.) which encourage creation, retention and dissemination of knowledge. Therefore, there is an urgent need to analyze agile software development practices from KM perspective. Many covert and overt factors are identified in applying agile practices in software development organisations. Different knowledge creation and management theories are analyzed from agile perspectives and relationship is established among knowledge management and agile practices with a special focus on Indian software engineering organisations.

Keywords: Agile software development, Knowledge management, Scrum, Extreme programming

1. Introduction

Technology progresses too fast, requirements *change at rates that swamp traditional methods* (Highsmith et al., 2000) and customers are no longer available to state their needs up-front, while, at the same time, expecting more from their software. As a consequence, several consultants have independently developed methods and practices to react to the inevitable change they were experiencing. These practitioners had their own philosophies about how software should be developed. However, all of them advocated close collaboration between software development and business teams, as opposed to silo development by software teams; face-to-face communication, as opposed to over-stress on written documentation; frequent delivery of segment of working software, as opposed to final delivery of the complete product at the end; accepting changing requirements, as opposed to defining fixed requirements. These principles (Fowler 2002) underlie the philosophy of agile software development (ASD). 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.

What is the meaning of being agile? Jim Highsmith enunciates that being agile means being able to deliver quickly, change quickly, and change often (Highsmith et al., 2000). In agile methods, people play a driving role in the success of the project and lot of short-time meetings are conducted for knowledge sharing and for the random change in the project, if required. Methodologists argue that working software without documentation is better than non-working software with a huge amount of documentation (Koskela and Teknillinen, 2003). There is no universally accepted definition of agility. Agility is dynamic, context-specific, aggressively change embracing, and growth-oriented (Goldman et al., 1995). The core concept in agile is quick response to change (Cockburn and Highsmith, 2001). A description of the various agile principles is given in the Agile Alliance (2004).

2. Review of Literature

Conboy and Fitzgerald (2004) carried out a review of the literature on agility across several disciplines and provided a broad definition of agility as the continual readiness of an entity to rapidly or inherently, proactively or reactively, embrace change, through high quality, simplistic, economical components and relationships with its environment. Despite the differences, all definitions of agility emphasize the speed and flexibility as the primary attributes of an agile organization (Gunasekaran, 1999). Schuh (2004) presents précis of agile development by stating that agile practices are not new, what is different and original about the agile approach is that the agile alliance has published these practices, fused them with core values about people and project environments and stated the way to build software better. Publication of the manifesto did signal industry acceptance of agile philosophy. Agile methodologies are made up of values, principles and practices. While agile practices may differ a little according to specific agile methodologies, there exist fundamental agile practices that are based on the four agile values and twelve principles and are common to all agile methodologies (Agile Alliance, 2004). Table 1 highlights some of the agile methodologies used frequently in literature along with the published resource work.

Table 1: Summary of Agile Methods by Earliest Date of Publication

Sr. No	**Agile Method**	**Acronym**	**Primary Source**	
			Journal Article	**Book**
1.	Dynamic Systems Development method	DSDM		Stapleton (1997)
2.	Crystal method	Crystal		Cockburn (1998); Cockburn (2002)
3.	RUP (Configured)	Dx		Martin (1998)
4.	Extreme Programming	XP	Beck (1999)	Beck (2000)
5.	Adaptive Software Development	ASD		Highsmith et al. (2000)
6.	Scrum	Scrum		Beedle, Devos, Sharon, Schwaber and Sutheriand (1999)
7.	Pragmatic Programming	PP		Hunt and Thomas (2000)
8.	Internet Speed Development	ISD	Cusumano and Yoffie (1999)	Baskerville and Pries Heje (2001)
9.	Agile Modeling	AM		Ambier (2002)
10.	Feature Driven Development	FDD		Paimer and Feising (2002)
12.	Lean Development	LD		Charette (2002); Poppendiek and Poppendiek (2003)

Values in agile manifesto give purpose to software development, complement each other, and are aligned with life goals (for example, putting more value on development of working software instead of writing comprehensive documentation). Principles are more general and they may clash (for example, cost versus quality, the principle is to maintain low cost and high quality). Practices are less flexible, they bring accountability, and they take the purposes depicted in the values to real practice (for example, pair programming improves the level of individual interactions within a team) (Beck and Andres, 2004). We further analyze agile manifesto according to the dependencies of the values and principles. The dependencies of agile manifesto values and principles are summed up in the Table 2.

Table 2: Enslavement of Principles on Agile Manifesto

Agile Manifesto		**Agile principles**											
Primary	**Secondary**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**
Individuals and interaction	processes and tools	X			X	X	X		X				X
Working software	comprehensive documentation	X		X				X		X	X	X	
Customer collaboration	contract negotiation	X	X		X				X				
Responding to change	following a plan	X	X	X				X					X

Although agile methodologies concur with the current software development practice, they are not all suitable for all phases in the software development life-cycle. Abrahamaaon et al. (2002) explain different phases of software development that are supported by different agile methods. Each method is divided into three blocks. The first block indicates if a method offers support for project management. The second block identifies whether a process is described within the method. The third block indicates whether the method describes the practices, activities to be followed and used. A gray color in a block indicates that the method supports the life-cycle phase and a white color indicates that the method does not provide detailed information. As shown in the Figure 1, these practices lack approaches that support software development, except RUP and DSDM which do not require any outside support.

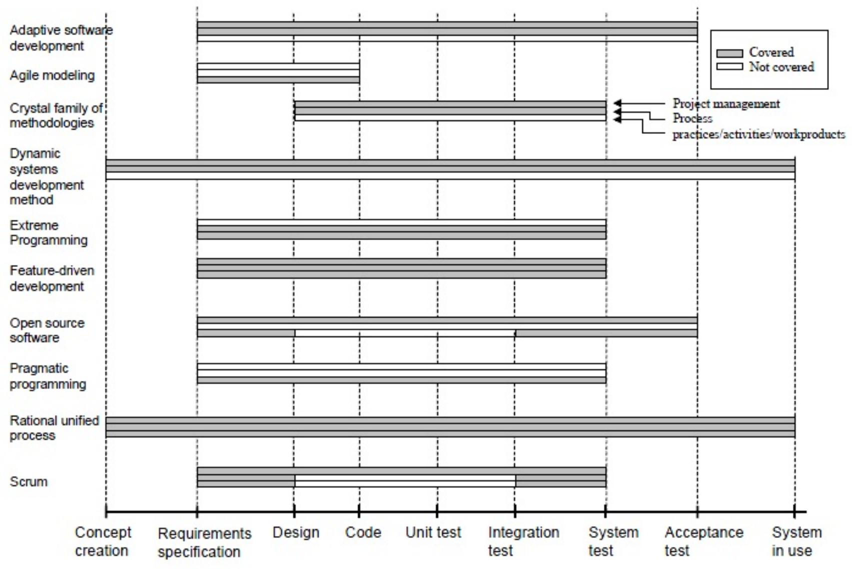

Figure 1: Life Cycle support of Various Agile Methods (Adopted from Abrehamsson et al. (2002)

Qurner and Henderson (2008) also examine product engineering process of software development. They state that product engineering process can be further divided into development process and project management process. Authors look into some of the agile methodologies and try to map their practices with development and project management processes. Table 3 shows the practices followed by different agile practices and category in which they fall.

Table 3: Enslavement of Agile Practices on Software Process

Software Process	**XP**	**Scrum**	**FDD**	**ASD**	**DSDM**	**Crystal**
Development Process	1.Short releases 2.Metaphor 3.Simple design 4.Testing 5. Refactoring 6.Pair programming 7.Collective ownership 8. Continuous integration 9. On- site customer	1.Scrum team 2.Product backlog 3. Sprints 4.Sprints reviews	1. Domain object modelling 2. Developing by feature 3.Indivdual class ownership 4. Inspections 5.Regular builds	1. The project mission development 2.Developing by components 3.Collaborative teams 4. Joint application development 5. Customer focus group reviews 6. Software inspection	1. Active user involvement 2. Empowered teams 3.Frequent product delivery 4. Fitness for business purpose 5. Iterative and incremental development 6. Reversible changes 7.Requirements are based at high level 8.Integrated testing 9. Collaboration and cooperation among stakeholders	1.Staging 2. Holistic diversity and strategy 3.Parallelism and flux 4.User Viewings 5. Revisions and reviews
Project Management Process	1.The planning game	1.Scrum master 2. Sprint planning meeting 3. Daily scrum meeting	1.Reporting/ Visibility of results	1. Adaptive cycle planning 2. Adaptive management model	Not specified	1.Monitoring of progress
Software Configuration control process /support process	Not Specified	Not Specified	1.Configuration management	Not Specified	Not Specified	Not Specified

Process management	Not Specified	Not Specified	Not Specified	1.Project post-mortem	Not Specified	1,Reflection workshops methodology tuning

In software development, knowledge is considered as the most important asset. Hansan et al. (1999) state that in practice, most organizational KM strategies are either codification strategies or personalization strategies. Lot of studies in literature have been found which show that agile software development practices use personalization strategies for managing the critical knowledge. Our arguments are with reference to the findings of Boehm and Turner (2004) as well as Paulk (2002) that compare plan-driven and agile software development and declare that plan-driven approaches generally prefer codification of knowledge while agile approaches mainly attempt to cultivate tacit knowledge. Similarly, Robinson and Sharp (2004), from their empirical studies on xp programming teams, conclude that respect and trust are important prerequisites for the successful implementation of knowledge sharing through conversation. Wendorff and Apshvalka (2005) in their research present the personalization strategy used in agile software development.

3. Research Methodology

Proponents of agile acknowledge the use of technology for successful implementation of personalization strategy of agile practices is distributed environment. It is vital to test this aspect from Indian perspective because most of the Indian software development industry is working with agile methodologies in distributed environment. Organisations are chosen on the basis of Goode's (2001) recommendations to measure size of the organisations by number of employees working in that firm. Stratified sampling technique is used for conducting survey on Indian industry. Therefore, software engineering (SE) organisations have been classified in five broad categories according to their size in terms of their employee strength, i.e. Very large-, Large-, Medium-, Small- and Very small- sized organisations. The SE organisations identified as 'very small' companies (having up to 50 employees) have been clubbed with 'small' (50-500 employees) to make a meaningful group of small companies. Furthermore, SE organisations identified as 'very large' companies (having more than 100,000 employees) have been combined with 'large' SE organisations (having employees between 5001 and 100,000) because the number of 'very large' companies in India is very less. Only registered companies with NASSCOM (National Association of Software and Service Companies) are included in the survey. Survey conducted in the study includes a mix of SE organisations based on functional specialization, i.e., organisations developing software alone, organisations providing only consultancy services, and organisations that perform both the functions.

Survey questionnaire was presented to 340 professionals working at different levels in Indian software engineering industry. Frequency count and weighted average score (WAS) are computed from the data and conclusions are drawn accordingly as per the recommendations of Sharma (2011).

$$WAS= \sum WX/\sum X$$

Where $\sum WX$ is total sum of weights assigned to responses. $\sum X$ is sum of number of responses. Five point Likert scale ranging from -2 to 2 (-2= strongly disagree, -1= disagree, 0- neutral, 1= agree, 2 strongly agree) is used for measuring the perceptions of the respondents.

4. Results and Discussion

A set of questions were presented to the respondent organisations to know the technological support they were providing to their employees to implement personalization of knowledge, as recommended by Hansan et al. (1999). Respondent organisations were asked about the technological support they were using for knowledge sharing in distributed agile teams. We calculate chi square, contingency coefficient and WAS for each dimension. Outcomes of the survey are presented in the following sections.

4.1 Project Planning Tools help in Sharing Real-time Status

A consensus was found when organisations were asked if they were using any project planning tool in distributed agile environment for managing their project work. All the organisations agreed that they were using tools for project management across different dimensions of the agile organisations (c.f. Table 4). Only difference is found in case of fully-agile and partially-agile companies, 71.3% of fully agile organisations use project management tool as compared to 54.9% of partially-agile companies.

Table 4: Project Planning Tools used to Share a Real-time Status

Group/Sub-group	**SD**	**D**	**N**	**A**	**SA**	**WAS**	**Statistics**
Size of Company							
I	3.8	0	32.4	36.9	26.9	0.83	χ^2= 31.537 C= 0.290
II	0	0	36.4	29.1	34.5	0.98	
III	8.6	2.9	21.4	51.4	15.7	0.63	
Core Area							
Product Development	3.6	0	38.9	35.3	22.2	0.73	χ^2= 19.69 C= 0.234
Consultancy	0	0	33.3	26.2	40.5	1.07	
Both	4.6	1.5	21.4	43.5	29.0	0.91	
Type of Company							
Fully Agile	1.1	0	27.2	36.2	35.1	1.05	χ^2= 21.88 C= 0.246
Partially Agile	6.6	1.3	36.2	38.8	17.1	0.59	
Software Industry	3.5	0.6	31.5	37.4	27.1	0.84	

* significant at 5% (p = 0.05) ** significant at 1% (p = 0.01)

WAS score of more than 1 in case of consultancy organisations confirms the fact that these organisations use this technique more than the product development organisations or organisations working in both the domains. Similarly, WAS score of 1.05 of fully-agile companies clearly shows the dominance of tools in distributed environment for agile teams.

4.2 Maintaining Code Repository in Cloud for Sharing Source Code

Maintaining the source code for distributed agile teams is also a hindrance as agile teams have to implement collective code ownership. So respondents were asked how they manage their source code. WAS score of 1.12 of consultancy firms shows that they prefer using source code in the cloud to maintain collective code-ownership. Similar trends were found in fully- and partially-agile organisations. WAS score of 1.08 of fully-agile organisations shows that these organisations mostly used cloud for the storage of their source code as compared to partially-agile organisations. Small organisations do not show complete agreement to this practice as around 28% of the respondents were neutral about this practice (cf. Table 5).

Table 5: Sharing of Source Code by Maintaining Single Repository in Cloud

Group/Sub-group	**SD**	**D**	**N**	**A**	**SA**	**WAS**	**Statistics**
Size of Company							
I	0	10.6	21.2	35.6	32.5	0.90	χ^2= 34.988 C= 0.305
II	0	19.1	17.3	17.3	46.4	0.91	
III	5.7	14.3	28.6	24.3	27.1	0.53	
Core Area							
Product Development	2.4	15.0	22.8	24.0	35.9	0.77	χ^2= 25.19 C= 0.263
Consultancy	0	0	35.7	16.7	47.6	1.12	
Both	0	17.6	15.3	35.1	32.1	0.82	
Type of Company							
Fully Agile	1.1	4.3	22.9	29.3	42.6	1.08	χ^2= 35.175 C= 0.306
Partially Agile	1.3	26.3	19.7	25.0	27.6	0.51	
Software Industry	1.2	14.1	21.5	27.4	35.9	0.83	

* significant at 5% ($p = 0.05$) ** significant at 1% ($p = 0.01$)

Around 55% of the overall organisations are using this practice with WAS of 0.83. Collective code ownership is a prominent practice of agile software development methodologies. Organisations were asked if they share source code in distributed agile teams by maintaining single repository in the cloud. 68.1% of size I organisations agreed that source code should be stored in cloud whereas only 51.4% of size III organisations agreed with the statement. Sharing source code through cloud decrements as we move down to organisations depending upon its size. As per core area, 67.1% of organisations working in both (product development and consultancy) agreed that cloud is used for sharing of source code. More than 71% of fully-agile organisations used cloud for sharing of source code whereas 27.8% of partially-agile organisations disagreed with this statement.

4.3 Overlapping of Development Hours for Synchronous Communication

To bridge time zone differences, it is always recommended to have overlapped working hours for distributed agile teams so that teams can communicate easily. Organisations were asked if they follow this practice. Table 6 confirms that size I and II organisations agree that this practice is used for synchronous communication between onshore and offshore teams, whereas 30% of size III organisations do not use overlapping of working hours. 67.1% of product development organisations use overlapped working hours, whereas 21.4% consultancies do not use overlapping of working hours. 81.9% of fully-agile organisations use this practice for synchronous communication and on the other hand 27.6% of partially-agile organisations reject this practice. WAS of 1.08 of fully-agile organisations gives us an idea as how SE organisations are implementing synchronous communication between onshore and offshore teams. On the other hand, WAS of 0.55 tells the partially-agile organisations do not overlap working hours for synchronous communication. Around 70% of software organisations are using this practice for synchronous communication which is also described by WAS 0.84.

Table 6: Overlapping of Development Hours

Group/Sub-group	**SD**	**D**	**N**	**A**	**SA**	**WAS**	**Statistics**
Size of Company							
I	0	10.6	14.4	43.1	31.9	0.96	χ^2= 41.997 C= 0.332
II	0	12.7	12.7	40.9	33.6	0.95	
III	5.7	24.3	25.7	11.4	32.9	0.42	
Core Area							
Product Development	2.4	17.4	13.2	43.7	23.4	0.69	χ^2= 37.933 C= 0.317
Consultancy	0	21.4	26.2	4.8	47.6	0.79	
Both	0	7.6	16.8	35.9	39.7	1.08	
Type of Company							
Fully Agile	1.1	4.3	12.8	48.9	33.0	1.08	χ^2= 52.06 C= 0.364
Partially Agile	1.3	26.3	20.4	19.7	32.2	0.55	
Software Industry	1.2	14.1	16.5	35.9	32.6	0.84	

* significant at 5% ($p = 0.05$) ** significant at 1% ($p = 0.01$)

4.4 Collaborative Tools to Mimic face-to-face Communication

Collaborative tools are also used to mimic face-to-face communication. WAS of 0.92 and 0.97 by Size I and Size II organisations respectively reveals the dominance of collaborative tool, especially in fully-agile organisations where WAS is 1.03 (cf. Table 7).

Table 7: Collaborative tools to mimic face-to-face communication

Group/Sub-group	**SD**	**D**	**N**	**A**	**SA**	**WAS**	**Statistics**
Size of Company							
I	3.8	3.1	18.1	46.9	28.1	0.92	χ^2= 36.384 C= 0.311
II	0	6.4	29.1	25.5	39.1	0.97	
III	11.4	11.4	24.3	28.6	24.3	0.43	
Core Area							
Product Development	3.8	3.1	18.1	46.9	28.1	0.93	χ^2= 25.44 C= 0.264
Consultancy	0	6.4	29.1	25.5	39.1	0.97	
Both	11.4	11.4	24.3	28.6	24.3	0.43	
Type of Company							
Fully Agile	1.1	3.7	21.8	38.3	35.1	1.03	χ^2= 16.044 C= 0.212
Partially Agile	7.9	8.6	24.3	33.6	25.7	0.61	
Software Industry	4.1	5.9	22.9	36.23	30.9	0.84	

* significant at 5% ($p = 0.05$) ** significant at 1% ($p = 0.01$)

Organisations were asked whether they use collaborative tools for communication in distributed environment. Almost all the organisations agreed to the statement that they use collaborative tools to mimic face-to-face communication except Size III organisations where 22.8% of the organisations do not deploy any collaborative tool to mimic face-to-face communication. Overall 66% of the software industry accepts that they use collaborative tool.

5. Conclusion

This paper is a pivotal step in order to understand agile from knowledge management perspective. Many practitioners argue that KM practices are embodied into agile practices which provide methodology an upper edge on traditional software engineering practices. We find that rather than codification, agile practices emphasize on personalization of knowledge, i.e. they rely more on tacit knowledge management. Technological support is examined from knowledge management perspective and it is found that technological support is vital for implementation of personalization strategy for knowledge management in agile organisations. Overall, we have an agreement with other researchers in the agile domain that agile emphasize on personalization of KM and technological support is essential for the implementation of this strategy.

References

Abrahamsson, P., Salo, O., Warsta, J., Ronkainen, J. (2002), "Agile software development methods: Review and analysis, *VTT Publications*, (478)

Ambler, S. (2002), "Agile Modeling: Effective Practices for EXtreme Programming and the Unified Process". *John Wiley & Sons*

Baskerville, R., Pries-Heje, J.(2001), "Racing the E-Bomb: How the Internet Is Redefining Information Systems Development Methodology". In B. Fitzgerald and N. Russo & J. DeGross (Eds.), *Realigning Research and Practice in Is Development: The Social and Organisational Perspective*, New York, Kluwer, pp. 49-68 (2001)

Beck K. (2000), "Extreme Programming Explained: Embrace Change", *Addison Wesley*

Beck, K. (1999), "Embracing change with extreme programming", *Computer,* 32(10), 70-77

Beck, K., Andres, C. (2004), "Extreme Programming Explained: Embrace Change", *Addison–Wesley Professional*

Beedle, M., Devos, M.,Sharon, Y., Schwaber, K., Sutherland, J. (1999), "Scrum: A Pattern Language for Hyperproductive Software Development". *In: Pattern Languages of Program Design*, 4, 637-651 Harrison, Ed. Boston: Addison-Wesley

Boehm, B. and Turner, R. (2004), "Balancing Agility and Discipline: A guide for the perplexed", *Addison–Wesley, USA*, first edition, 165–19

Charette, R. N. (2002), "Foundations of Lean Development: The Lean Development Manager's Guide". 2, *The Foundations Series on Risk Management* (CD). Spotsylvania, Va.: ITABHI Corporation

Cockburn, A., Highsmith, J. (2001), "Agile software development: the people factor", *IEEE Computer,* 34(11), 131–133

Cockburn, A. (1998), "Surviving Object-Oriented Projects", *Addison-Wesley*

Cockburn, A. (2002), "Agile Software Development", *Addison-Wesley*

Conboy, K., Fitzgerald, B. (2004), "Toward a conceptual framework of agile methods: A study of agility in different disciplines", *Proceedings of the 2004 ACM Workshop on Interdisciplinary Software Engineering Research*, 37-44

Fowler, M. (2002), "The Agile Manifesto: where it came from and where it may go", http://martinfowler.com/articles/agileStory.html

Goldman, S.L., Nagel, R.N. and Preiss K. (1995), "Agile Competitors and Virtual Organizations: Strategies for Enriching the Customer", *Van Nostrand Reinhold*

Goode, S. (2001), "Organisational size metrics in IS research: a critical survey of the literature 1989-2000". *In: Finnie, G., Cecez-Kecmanovic, D., Lo, B. (eds.). In: Proceedings of 12th Australasian Conference on Information Systems*. 257-68

Gunasekaran, A. (1999), "Agile manufacturing: a framework for research and development", *International Journal of Production Economics*, 62, 87–105

Hansen, M.T., Nohria, N., Tierney, T. (1999), "What is your strategy for managing knowledge?" *Harvard Business Review*. 77(2), 106 - 116

Highsmith J., Orr K., Cockburn A. (2000), "E-Business Application Delivery", www.cutter.com/freestuff/ead0002.pdf, 4-17

Hunt, A., Thomas, D. (1999), "Pragmatic Programming". *Addison Wesley, First Edition*

Koskela, teknillinen tutkimuskeskus, V. (2003), "Software configuration management in agile methods", *VTT Technical Research Centre of Finland*

Palmer, S.R., Feising, J.M. (2002), "A Practical Guide to Feature-Driven Development", *Prentice Hall*, (2002)

Paulk, M.C. (2002), "Agile Methodologies and Process Discipline", *CROSSTALK, The Journal of Defense Software Engineering*, 15(10), 15-18

Poppendieck, M., Poppendieck, T. (2003), "Lean Software Development: An Agile Toolkit" *Addison-Wesley Professional*

Qumer, A., Henderson-Sellers, B. (2008), "An evaluation of the degree of agility in six agile methods and its applicability for method engineering", *Information and Software Technology*, 50(4), 280–295

Robinson, H. and Sharp, H. (2004), "The Characteristics of XP Teams", *In Eckstein, J. and Baumeister, H. (Editors) Proceedings of the 5th International Conference on Extreme Programming and Agile Processes in Software Engineering (XP 2004), Springer*, 139-147

Schuh, P. (2004), "Integrating Agile Development in the Real World", *Charles River Media*, Massachusetts, USA, 1–6

Sharma, N. (2011), "Design of Experience Base Model for Software Process Improvement". *Doctoral Thesis, Punajbi University, Patiala* (2011)

Stapleton J. (1997), "DSDM: The Method in Practice", *Addison-Wesley Longman Publishing Co., Inc*

Wendorff, P., Apshvalka, D. (2005), "The Knowledge Management Strategy of Agile Software Development", *Proceeding of: 6th European Conference on Knowledge Management*, University of Limerick, Ireland