The Knowledge Engineering Review, Vol. 29:4, 496–511.
doi:10.1017/S0269888914000198

Conceptual modeling for knowledge management to support agile software development

AMRITESH and SUBHAS C. MISRA

Department of Industrial and Management Engineering, IIT, Kanpur 208 016, India;
e-mail: amritesh@iitk.ac.in, subhasm@iitk.ac.in

Abstract

Agile software development (ASD) has emerged as a result of consolidated values proposed under the lightweight methods of software engineering. Despite bearing some criticisms, the initial deployment and results observed in the practice environment represents its increasing domination over the traditional software development practices. Any ASD method, in particular, requires knowledge-intensive practices and typically employs multi-disciplinary expert team working extended periods of time for weeks on a nearly continuous basis. A huge amount of tacit knowledge creation and exchange happens in the entire process over the project lifecycle, which attracts the attention of research in the domain of knowledge management (KM). In this paper, first, we have mapped the agile values and agile principles, and in its support, we have argued upon and the need for integrated KM infrastructure and proposed a KM model that can be employed within the organization. We have also developed a conceptual framework for knowledge sharing and learning for the individual practitioners for the sustainability of agile team. We attempt to create an organizational learning framework for knowledge creation and exchange among the involved entities in a collaborative practice environment.

1 Introduction

Software development in general and agile practices in particular, have been accepted as a knowledge-intensive process (Corbin *et al.*, 2007), in which knowledge in various forms keeps circulating through every stages of the process and all across the project team including analysts, developers, and customers. The agile software development (ASD) approach to software development was initially advocated by experienced practitioners to address the commonly occurring problems such as running over budget (Jorgensen & Moløkken-Østvold, 2006), being behind schedule, and producing output of poor quality. Such problems are debated to be known as software crisis (Agerfalk & Fitzgerald, 2006) and requires to develop a solution framework. In order to find solutions to those problems, the software development paradigm shifted to adopt new set of strategies emphasizing upon communication, collaboration, and teamwork (Agerfalk & Fitzgerald, 2006). Equipped with such practice components to address those problems, agile approaches have emerged and are said to have the potential to adapt to the volatile requirements by self-organizing multi-disciplinary teams and consistent customer collaboration (Cockburn, 2000). Agile methods are predominantly executed by face-to-face communication between team members and onsite customers. Intensive team interaction in agile methods is always accompanied by extensive brainstorming among the individuals, resulting in huge amount of tacit knowledge creation and sharing. As agile methods discourage extensive documentation of the development process, most of the knowledge created during the development process resides within the people's mind and possesses risk of being lost when they leave the organization. The acquired knowledge always reside within the mind, action, and behavior of the agile team members, which indicates that their knowledge and experience stays with the

organization as long as they stay practicing or providing consultancy support in the organization. Traditional knowledge management (KM) practices were aimed at protecting the tacit knowledge by capturing and codifying it in the documents, standards, routines, process manuals, experience stories, and patented documents, and thereafter putting them into the organizational knowledge repository. However, in the context of ASD, there is a need to adopt a new approach of KM where knowledge predominantly exists in tacit form and observed in the collective actions of the agile team members who act as dynamic knowledge repository.

To reap the benefits of agile methods for those firms who rely completely upon traditional software development practices, it may be possible to adopt agile practices on partial and phased manner for different projects, or for different subcomponents of the same project running in parallel. A review (Bjornson & Dingsoyr, 2008) on ASD quotes that agile methods are not competitors of the tradition methods; rather both the methods can have a symbiotic relationship, in which factors such as the number of people working on a project, application domain, criticality, and innovativeness determine which process to select. It further says that the project managers can carefully review the characteristics of several agile practices and make their choices for implementation according to the project requirements. A critique on agile methods limits their applicability and scalability to distributed development environments (Turk *et al.*, 2002), which support our assertion that it would be a better strategy to adopt both the methods, considering the nature and limitations of the project components. In such a case, a separate KM framework is required, which can be integrated and synchronized with the traditional KM practices in order to demonstrate business value.

The rest of the paper is organized into seven sections. Section 2 explores the predominantly existing KM in traditional software development process. Section 3 maps the core values and principles emphasized under agile methods and tabulate the parameters to evaluate the agility of any software development process. Section 4 discusses the forms and value of knowledge and explores into contextual meaning of knowledge and its significance in ASD method. Section 5 maps the forms of knowledge to the agile practices and distinguishes between different forms of knowledge possessions in this context. Section 6 discusses the significance of knowledge sharing and various modes of learning. It further develops a learning model for the agile project team. Section 7 elaborates upon the integrated KM practices in the agile practices and finds its facilitation support with knowledge network and experience factory. Section 8 concludes the paper, emphasizing the viability and sustainability concerns of the developed knowledge sharing and learning support models.

2 Knowledge management in software development process

The software development process renders knowledge-based activities, particularly creation and application of both tacit and explicit forms of knowledge at every stage (Aurum *et al.*, 2008). The process is accompanied by continuous learning, accumulation of experience over the time, and its application in the work process. KM is an entirely different and a new discipline of research that has found its extensive applications in software engineering. In the Introduction chapter of the book *Challenges and Issues in Knowledge Management* (Buono & Poulfelt, 2005), the activities of KM in general is said to pass through different generations having different areas of focus. The first generation have assumed knowledge as a possession and the KM activities were primarily focused toward capturing the knowledge, and its dissemination with the help of technological tools such as information system, and knowledge repositories. In the second generation of KM, the practices are focused to observe and understand knowledge creation, and transfer through social interactions, *communities of practices* (CoP, Wenger, 1996), and complex human system. In this generation, a contemporary study (Carter & Scarbrough, 2001) has argued to map and integrate KM with human resource management, and identified learning as the intersecting domain between these two different disciplines.

The commonly known KM activities include knowledge creation, knowledge capture, knowledge storage, knowledge dissemination, knowledge sharing, and knowledge usage. Aurum *et al.* (2008) investigated the KM practices in Australian software development organizations by categorizing the KM activities in seven conceptually different components: knowledge creation, knowledge acquisition, knowledge identification, knowledge adaptation, knowledge organization, knowledge distribution, and

knowledge application. In context of ASD, the imperative considerations are how knowledge is personalized, applied, and shared.

A lot of research has been done to explore and evaluate the application of KM practices in various strategic forms to facilitate traditional software development practices (Corbin *et al.*, 2007; Aurum *et al.*, 2008; Bjornson & Dingsoyr, 2008). A review on 'Knowledge management in software engineering' (Bjornson & Dingsoyr, 2008) differentiates the research and applications on KM from two separate vantage points: technocratic school and behavioral school. In this review framework, the technocratic school is further trisected into three different schools: systems, cartographic, and engineering. The practices of KM in this school focuses on codification of tacit knowledge, employing knowledge repositories, development of technical infrastructure for sharing tacit and explicit knowledge, and promotion of learning for the process improvement. The behavioral school explores into the organizational and strategic domains of KM. The organizational school focuses on essence of formal and informal networks for knowledge sharing, in order to build knowledge community. The strategic school views KM as a competitive strategy and focuses on the learning process for building *intellectual capital*.

2.1 Motivation behind the study

The research on ASD is still in the emerging state and there is almost no work done to relate it to the domain of KM research. A research finding (Bjornson & Dingsoyr, 2008) discovers some possible applications of KM into ASD. It says that the cartographic school of KM, which focuses on creation of Yellow Pages as knowledge directories, can be applied to ASD because of its low-cost technical infrastructure. Another work in this direction (Chau & Maurer, 2004) provides a framework for knowledge sharing in ASD and illustrates a technological infrastructure that can help in synchronous as well as asynchronous communication for knowledge sharing among co-located as well as distributed agile team members. However, it does not elaborate how such a system can be developed further to support integration of KM practices of traditional and agile methods in an organization where management uses both the software development processes according to the nature and size of the project. To fulfill this research objective, it is essential to explore deeper into the agile features of the process, and forms of knowledge present over there in various forms. This can further lead to develop a KM strategy for imparting sustainability to the agile methods that can further be integrated with the KM framework of the legacy system.

3 Characterizing features of ASD process

The agile methods have proved to be more effective, efficient, productive, and reliable as compared with the traditional methods (Bjornson & Dingsoyr, 2008). This creates curiosity to dig deeper into the fundamental features of this methods and find the inherent values. There are certain distinguishing general characteristics of ASD, which makes it different from the traditional software development methods. The practitioners advocating agile approach for software development have agreed upon certain common principles and values[1]. We present the mapping of the commonly accepted 4 agile values to the corresponding 12 agile principles (refer Table 1).

Apart from the above-mentioned general characterizing features, Qumer and Henderson-Sellers (2008b) have developed a detailed framework containing characteristic features to measure the degree of agility to compare and analyze different agile methods. This is known as 4-DAT analysis framework that has been successfully adopted and applied to compare and analyze the agility within various ASD methods and also within traditional software development methods (Qumer & Henderson-Sellers, 2006, 2008a, 2008b). This framework has also been applied to evaluate agile method adoption and process improvement models in ASD (Qumer *et al.*, 2007; Qumer & Henderson-Sellers, 2008a, 2008b). This framework classifies the agile factors into four broad dimensions: method scope, agility characterization, agile value characterization, and software process characterization. These parameters, apart from evaluating the degree of agility, also act as perspectives to map and design the supportive features for KM in this domain.

[1] Agile principles and values are available on the website `http://www.agilemanifesto.org`

Table 1 Mapping the agile values and principles

Agile values	Agile principles
1. Individuals and interactions over processes and tools	1. Build projects around motivated individuals. Give them the environment and support they need, and trust them to get the job done. 2. The most efficient and effective method of conveying information to and within a development team is face-to-face conversation. 3. The best architectures, requirements, and designs emerge from self-organizing teams.
2. Working software over comprehensive documentation	4. Deliver working software frequently, from a couple of weeks to a couple of months, with a preference to the shorter timescale. 5. Our highest priority is to satisfy the customer through early and continuous delivery of valuable software. 6. Working software is the primary measure of progress. 7. Simplicity—the art of maximizing the amount of work not done—is essential.
3. Customer collaboration over contract negotiation	8. Business people and developers must work together daily throughout the project. 9. Agile processes promote sustainable development. The sponsors, developers, and users should be able to maintain a constant pace indefinitely.
4. Responding to change over following a plan	10. Welcome changing requirements, even late in development. Agile processes harness change for the customer's competitive advantage. 11. At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behavior accordingly. 12. Continuous attention to technical excellence and good design enhances agility.

4 Recognizing the form and value of knowledge

4.1 Forms of knowledge

Knowledge, by definition, has terminological ambiguity and literature has tried to classify it into two epistemologically distinct forms: tacit and explicit knowledge, according to the ease of articulability (Nonaka & Konno, 1998; Novak & Wurst, 2005). The ease of codification and convertibility of knowledge into written or documented form makes it closer to the definition of explicit knowledge, which can be structured and stored; while the other form of knowledge is difficult to codify and document, and categorized under the definition of tacit knowledge, which generally remains in unstructured and unexpressed forms. Written documents, scientific formulae or specification, process manuals, and technical writings come under the domain of explicit knowledge, which can be easily captured and transmitted formally and systematically. While the tacit knowledge resides in the minds of the people in the implicit forms of 'mental constructs', and reflects in the skill, behavior, culture, attitude, and experience. It is highly personal, hard to formalize, and difficult to communicate. Tacit and explicit forms of knowledge are often expressed as soft and hard forms, respectively (Hildreth & Kimble, 2002). One of the activities identified under the first generation of KM is found to capture the tacit knowledge and codify it into explicit form, and store it into the electronic knowledge repositories (Bjornson & Dingsoyr, 2008). One can conceive that explicit knowledge is derived from the tacit knowledge process by categorized context and specifications. The declarative context can help to understand the know-about phenomena, and answers the questions starting with what, which, and who. Procedural context can assist to understand the know-how phenomena (i.e. the process). Similarly, causal context is mapped to know-why phenomena, which explains the ways to understand reasons and justification. Conditional context is mapped to know-when phenomena and one can know the prerequisites for happening of any event. Relational context can be mapped to the know-with phenomena, which can help to understand different kinds of relationships among the participating entities. Thus, explicit knowledge can be derived from the implicit knowledge in a more comprehensible form, articulated and codified through defining the context and setting. An alternative argument exists against the codification strategy, which says that all the tacit knowledge can never be captured and converted into explicit form (Hildreth & Kimble, 2002).

The key properties of knowledge include: tacitness, dispersion, context specificity, transferability, reception or absorption, and complexity (Quintas *et al.*, 1997). From epistemological perspective, any knowledge is valid only within the context and settings in which it is acquired and learnt. Before applying the previously acquired knowledge into new conditions requires the knowledge of both the contexts, that is, context of knowledge acquisition and context of knowledge application. Therefore, knowledge of context is itself a separate kind of knowledge that is critical during the process of learning and practicing.

Any process is driven by planned systematic actions by the concerned actors at appropriate time. All the actions are governed and managed either by inherent capabilities, and/or the formal procedures documented by the organization. The human mind observes and interprets the surrounding in context of previously acquired concepts, values, and beliefs, which forms a part of knowledge, and stays with him in an embodied state, which is recognized as a mental construct. This mental construct acts as a framework to understand, evaluate, and acquire new knowledge at a much accelerated pace.

4.2 Value of knowledge

In a psychological perspective, knowledge can be considered as a prerequisite of human action and is created through changes in cognitive structure (Wissensmanagement Forum, 2003). Following the cognitive dimension, knowledge, as a static object, can be recognized as a state of mind within the individuals. At a certain level, state of mind turns out to be a capability to perform some task. A set of capabilities are recognized as a condition for accessing the resources and authoritative power. These are the forms of knowledge that relates to the tacit of soft side that remains in potentially unexpressed form, and not always manifested in codified forms. The tacit knowledge always reflects in the individual behavior, attitude, and practices. This is critical to the organizations in which behavioral interactions are considered as an important part of the work culture and collective action dominates over the structured individual roles. During the collective interaction, knowledge is owned collectively by a social group or an organization and recognized as a key asset responsible for the organizational economic growth. Knowledge is also considered as an organizational asset that requires strategic emphasis of management to take competitive advantage (Alavi & Leidner, 1999). Assets constitute the resource base of any firm and are employed in optimizing the production efficiency. Knowledge is also seen as a strategic resource (Probst *et al.*, 1999) and relates it to the process of learning, which can be leveraged to generate knowledge resource in order to gain competitive advantage. Resource perspective of knowledge requires short- term and long-term strategic planning as a part of management. The economic perspective identifies knowledge as a value-based technological or intellectual asset, which is either incorporated within any product or individual of the organization and constitutes a part of intellectual capital (Sullivan, 1999). The business value of intellectual capital is often intangible and its influence on the operational performance is observed for a prolonged period in the organization.

5 Knowledge possessions in agile software development

In agile practices, the software development team is supposed to be expert and multi-disciplined and should reveal higher states of knowledge level. Bohn (1997) describes the transition during gradually developing knowledge states about a process, which start with the level of complete ignorance about the process, and goes up to the level where one can have complete control over the process and optimize the performance level. These knowledge states can be mapped with the CMM levels of the process characteristics, where the optimized level of process is essentially driven by expert individuals and team.

The expert team of agile practices possesses mature and high-level skills to quickly perceive and analyze the project development environment. They can easily grasp the customer requirements and prepare the release and iteration plans. Similarly, developers and testers are also assumed to be the domain experts, so that they can easily and quickly adapt to the volatile requirements of customers. One can easily realize that in agile methods, there is high concentration of knowledge in the minds of practicing individuals in the form of their previously acquired skills. The agile team members possess knowledge of both technical and cognitive dimensions of tacit knowledge. The technical dimension constitutes technical

skills, for example, the language of communication and proficiency over using a technology, which are commonly known as know how, while the cognitive dimensions consists of mental models, schemata, ideals, values, and beliefs, which are deeply ingrained within the individual (Nonaka & Konno, 1998) and acts as a framework to evaluate, analyze, and grasp the new knowledge. Cognitive dimension of knowledge is also responsible for knowledge sharing behavior, and reflects into the participation toward collaborative culture of the organization. Cultivation of knowledge sharing and collaborative organizational culture are not only sufficient, but essential conditions for ASD process. Here, our prime consideration about KM remains on the soft knowledge. According to Hildreth and Kimble (2002), soft knowledge can include tacit knowledge, internalized experience, skills, domain knowledge, and knowledge embedded into cultural practices. Soft knowledge is relevant with the constructionist perspective, where knowledge is supposed to be constructed by social processes involving mutual interaction, performing upon collective tasks, etc. In ASD context, soft knowledge is composed of project development experiences (e.g. executing development iterations, software testing, reconfigurations, quality, etc.), coding skills (e.g. using syntax, logics, effective and efficient organization of subroutines, etc.), knowledge about customers preferences (software requirement specifications, negotiability of delivery plans, etc.) as customers are also a part of ASD team.

A large portion of the knowledge appears to be collective in nature and manifested in dynamic forms in the collective actions, for example, interaction during pair programming, doing requirement analysis, preparing delivery schedules and iteration plans, and refactoring. Some of these collective processes are driven by intensive brainstorming and interaction between the individuals engaged in their respective activities, and results into dynamic creation of knowledge in the organization. In an organizational perspective, CoP are said to offer an environment for its members for mutual interaction, as a result of which knowledge is created through situated learning, nurtured ,and sustained (Hildreth & Kimble, 2002). CoPs are an ideal example of a dynamic knowledge creation process, in which learning happens as a social process through interactive participation of knowledge actors (Wenger, 1996). The dynamic creation of knowledge by this process may result into a new form and quality of knowledge, which can be much more refined than the previously acquired knowledge by the individuals. This process of knowledge creation can drive innovation, which can rejuvenate the entire development process. From the perspective of ownership and control, this phenomenon can be considered as a dynamic possession of collectively created knowledge by the entire team during the time of creation, when ideas keep circulating among all the brains in order to reach a conclusive form. Performing requirement analysis, preparing delivery schedule, and finalizing test plans are such processes that indicate the dynamic possession of knowledge by the individual and group.

Following the above-mentioned phase, part of the product requirements, delivery schedule, and test plans are frozen, and based on that the software development process enters into the next phase. The partly frozen plans and schedules also constitute a part of the knowledge input for the next phase of the process. This knowledge can be considered under static possession by the entire team. Moreover, collective ownership of code during pair programming is also a form of collective possession of the knowledge that is considered in static form. A macro observance in this context can perceive that knowledge can be owned either individually or collectively or in both the states: static and dynamic, or in both into different proportions.

During the collective possession of static knowledge, the knowledge is able to be codified or documented and made available to the users. Such knowledge can be stored in the knowledge repositories for future uses, training, and learning purposes. On the other hand, the dynamic possession of knowledge appears to have a comparatively short lifecycle and its future usages are limited to the period until it remains in the users' mind. This knowledge remains in use only during the time it is being applied by the individual in the assigned tasks. For rest of the times, this knowledge remains potentially stored in soft form within the individual and remains under high risk of being lost unless it is documented and stored in some hard form, or is shared to the other members of the organization who personalize it.

5.1 Focussing on the key roles and knowledge-based tasks

In order to develop a KM system in ASD methods, appropriate considerations may be given to the key processes, and the key activities or roles that an individual performs. Despite of being similar in nature,

Table 2 Individuals participating in learning and knowledge sharing

	Roles and responsibilities of respective agile team			
Key roles	Extreme programming	SCRUM	Rational unified process	Dynamic systems development method
Manager/system analyst	Manager, coach, tracker	Management, SCRUM master, product owner	Business model reviewer	Visionary, technical coordinator
Developer	Programmer consultant	Scrum team	Course developer, tool smith	Developer
Tester	Tester	Scrum team	Course developer, tool smith	Senior developer
Customer	Customer	Customer	Customer	Ambassador user, advisor user, executive sponsor

Developed from Carter and Scarbrough (2001).

different agile methods, for example, extreme programming (XP), SCRUM, rational unified process (RUP), and dynamic systems development method (DSDM) employs different process structure and different compositions of roles and responsibilities for the individuals. A book *Agile Software Development Methods: Review and Analysis* (Abrahamsson *et al.*, 2002) presents a brief description of some of the agile methods that are in practice. On the basis of explanation of agile methods given in that book, we have categorized the significant individual roles of the agile team into four specific classes: customer, system analyst, developer, and tester. Each of these roles possess essential requirement of certain expertise and knowledge by the ASD team members to perform on the project. The soft knowledge as described in the previous section is collectively possessed by respective members playing specific roles. Apart from that, the manager also has his/her role in some of the agile methods such as XP and SCRUM. However, he/she does not play a direct role and remains latent in the process of knowledge creation though learning. Learning is essential not only during the working interaction and tacit knowledge exchange among the project team members, but also to provide support during introduction of any new technology in the organization. Moreover, this can further support in continuous process improvement to increase the overall product and process quality. Table 2 represents the classification of key functional roles for four different agile methods XP, SCRUM, RUP, and DSDM and classifies them into five broad categories. In addition, we have identified the key phases of those four agile methods and classified them into four broad categories: requirement analysis, system design and development planning, coding, and testing and maintenance, all accompanying tacit knowledge transfer in the ASD team. Table 3 represents the respective classifications.

6 Exploring conceptual models for knowledge sharing and learning

In Section 5, we discussed about the static and dynamic possessions of knowledge by the individual and team or organization. We also discussed about the significance of learning and knowledge sharing in the agile processes. In this section, we explore some knowledge transfer models that support knowledge creation through collective learning and knowledge sharing within and across the project teams.

The fundamental model of knowledge transfer is explained by Nonaka and Konno (1998) through a SECI model, where knowledge creation is represented as a spiraling process of interaction between tacit and explicit forms. SECI is a spiral process of four conceptually separate phenomena: Socialization (S), Externalization (E), Combination (C), and Internalization (I). Socialization is a phenomenon of experience sharing between individuals by spending time together and doing verbal communication. This phase is characterized by the process of knowledge co-creation happening as a result of individual interactions. At this stage, the authors say that learning happens by empathizing with colleagues and customers rather

Table 3 Key process of knowledge creation

	Agile method			
Major processes	Extreme programming	SCRUM	Rational unified process	Dynamic systems development method
Requirement analysis	Exploration	Planning	Inception	Feasibility/business study
System design and development planning	Planning	High-level design	Elaboration	Functional model iteration
Coding	Iteration	Sprint	Construction	Design and build
Testing and maintenance	Productionize maintenance death	Maintenance	Transition	Implementation

Developed from Carter and Scarbrough (2001).

than sympathizing with them. Externalization is a process of codification of the tacit knowledge and disseminating it into comprehensible forms through digital or physical mediums. Combination involves capturing of explicit knowledge and makes it more usable in different application domains. Internalization involves learning and application of explicit knowledge into organizational practices. The nature of knowledge flow within the agile teams comes under the process of socialization and internalization. In a scenario, where several agile teams are working on different projects or different components of the same project in an organization, then the kind of knowledge flow between those teams can be characterized under the process of combination. Lack of documentation restricts the agile method from undergoing the process of knowledge externalization.

An alternative literature on 'Review of learning 2.0' (Redecker, 2009) refers to the Semens knowledge flow cycle consisting of six similar steps: knowledge creation, co-creation, dissemination, communication, personalization, and application. The internalization phase of Nonaka and Konnos model combines two different phases personalization, and application of Semens model of knowledge flow. Actually, this is the focus area where learning happens through mutual interactions. Personalization is a way of learning that brings knowledge into dialog and reflection, which can be applied into practice, while application of knowledge also creates experience and tacit knowledge within the individual, which is also being personalized at the same time. This situation can be segregated and understood by the knowledge creation strategy having two different action orders: first is learn and apply, the second is applying and learn. These action orders can also be mapped with two different strategic learning situations just in case (JIC) learning and just in time (JIT) learning. The JIC strategy involves gathering knowledge from previous experiences of others, reading documents, and case studies. On the other hand, JIT learning involves acquiring experiential knowledge by practicing in real-time situations (Modesitt *et al.*, 1999; Brandenburg & Ellinger, 2003). The two modes of learning can be further compared with the STI and DUI modes of learning (Lundvall & Lorenz, 2007; Guo *et al.*, 2008). The DUI (doing, using, and interacting) mode is said to be opportunistic learning, which is experience based and relies on building a tacit understanding over know-how and know-who kind of knowledge. STI mode can be assumed close to systematic or theoretical learning, which is associated with the explicit knowledge, and builds a tacit understanding over know-what and know-why aspects of knowledge, which may be obtained through reading books, attending lectures, and accessing databases. The DUI mode is a localized learning that is normally observed in co-located communities as it happens to be in CoPs. Knowledge creation through this mode of learning is imperative to the ASD team. Pair programming, release, and iteration planning, preparing test plans, and refactoring are those key activities of tacit knowledge exchange, in which situated learning takes place within the co-located teams.

JIT or DUI is a kind of onsite learning that is always accompanied by a lot of experimentation by the inexperienced individuals. This can expose to risk of process failure, or bad quality output when applied in mainstream business. Therefore, this can be useful to the R&D sections of the organization in which

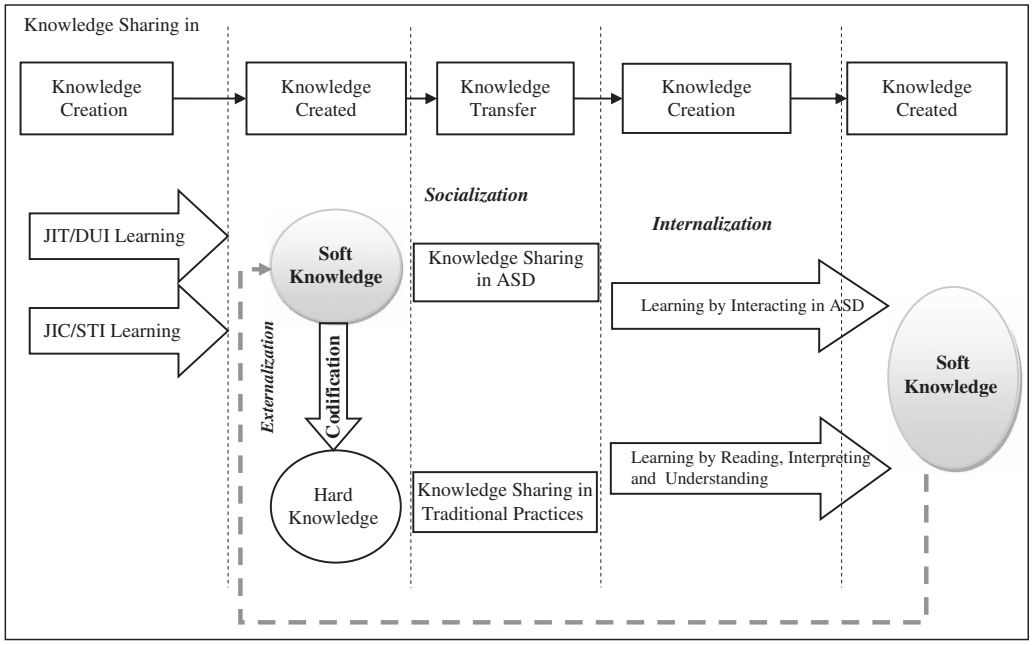

Figure 1 Conceptual model for knowledge sharing and learning

innovation is encouraged by providing space for absorbing the failures and newly learned knowledge is experimented at a small scale for its evaluation. However, in real-case scenarios, despite being favorable for promoting innovation, JIT or DUI learning can be a risky learning strategy for the agile team under action, and can adversely impact upon the entire process and product. On the other hand, JIC or STI mode of learning is accompanied by creation and usages of hard form of knowledge, which consumes more time and resources and cannot be a feasible learning strategy for sustaining the expertise of the agile team. Therefore, the requirement is to develop a different kind of learning strategy for the agile methods, which can facilitate experience capture and promote continuity of leaning in order to keep preparing intelligent members for the agile team. Both the kinds of learning results in valuable knowledge creation, and the created knowledge remain in the soft form. This knowledge can be converted into hard form and stored in the organizational knowledge repository for the future usages. However, all the soft knowledge cannot be converted into hard form, and partly still remain with the individual who had acquired it. Therefore, there may be a possible knowledge loss in this process of knowledge transfer. Instead of converting the hard form and then disseminate it for sharing and learning, the soft form of learned knowledge can also be directly shared to other by direct interaction and this process of knowledge transfer is much efficient than the previous form. The knowledge sharing during ASD happens through the similar way, by face-to-face interaction. The conceptual model for knowledge transfer and learning is presented in Figure 1.

This conceptual model explains the different phases of knowledge creation, transfer, and learning under ASD practices. Here, we are considering learning as a type of knowledge creation through interactive practices in the individuals mind, which is further incorporated in his/her task. The first phase of knowledge creation deals with the ASD team building. All members of the ASD team are supposed to be domain experts having rich skillset about the entire process. Their expertise comes from past learning and accumulated experiences. At this level, learning can happen in either of two modes: STI and DUI, or both. Thus, knowledge creation takes place and brought forward to the next phase, where knowledge remains as a (objectified) individual possession. The knowledge components at this level primarily includes past project experience and coding skills (e.g. using syntax and logics, organization of subroutines, etc.) that constitutes soft form of knowledge possession. This soft knowledge can be externalized into hard form by codification in case of traditional software development practice. As we are considering an integrated approach, where the firm moves a part of its development project to ASD team, same team members can keep moving between ASD and traditional development project groups. In this case, there is a possibility

to externalize and codify the experiences and knowledge of ASD practitioners as soon as he joins backs the traditional team. This externalized knowledge is captured by documentation that can be used for grooming a new ASD team with a better set of capabilities. The next phase is knowledge transfer through socialization. ASD team members remain in regular face-to-face interaction during the entire project development period and share their expertise as required. Each member of the team assists each other in performing the various phases of project progress. Knowledge sharing can also happen within the ASD team members by sharing the documented best practices when they are not on an ASD project, as it conveniently happens in tradition software development team. During internalization phase, ASD team members learn and incorporate the shared knowledge into their personal knowledge base and practices. During ASD project, learning normally happens in DUI mode and during off project period it happens in STI mode. This creates fresh soft knowledge within the ASD team. In this soft knowledge composition, the additional component brought is the customer's knowledge. As discussed earlier, ASD team also includes customers representative(s), which normally does not happen to be in the traditional software development teams. The knowledge created out of customer interaction (which includes understanding their language, development requirements, quality standards, flexibility in delivery schedule, reactions under development uncertainties, negotiability of quality, etc.) is the most critical element, which was the missing part of soft knowledge composition initially, now becomes a part of the fresh ASD team. This newly constituted soft knowledge base again acts as a rich source of base capability for ASD team that is connected to the demand side.

7 An integrated knowledge management approach

The main characterizing attributes of ASD process, which distinguishes it from traditional methods, are process efficiency and customer satisfaction. One of the methods adopted by agile teams to increase process efficiency is to reduce delivery time by excluding the process of extensive documentation, which incurs huge share of the entire software development time. Agile practices discourage comprehensive documentation and resist the externalization of knowledge into hard form. Therefore, it can work effectively as long as individual retains the expert knowledge, and stay working in the agile team of the same organization for longer periods. This phenomenon exposes to a big risk upon the sustainability of the agile process, if it would not be able to keep the experts in its project team for prolonged durations. High attrition rates of software development industry poses questions upon the sustainability of the same team with same experts who work on the project on a continual basis. Management needs to think upon the sustainability of the agile team, which can support the learning process going in parallel with the actual work process, so that the risk of loss of organizational knowledge is reduced to a minimum level. For an organization that shifts the part of its project process components from traditional to agile, or employing agile development process in parallel with the traditional processes according to the respective scopes, the strategic focus should be to employ an integrated KM infrastructure as well as an integrated knowledge sharing culture in the organization that supports continuous learning, experience accumulation, and transfer during the normal work process. The aim is to have a knowledge sharing environment within and among the agile teams, or between agile team and traditional team, in order to facilitate the flow of knowledge all across the organization. Learning for experience transfer also takes a central focus in this context.

For the traditional software development process, Aurum *et al.* (2008) has defined KM as a set of KM activities in order to create, store, transfer, and apply knowledge on a knowledge business value chain using appropriate technologies and cultural environments. Technology and culture have been seen as the two key aspects of this domain. Technology assists in KM in two fundamental forms. First, it helps to codify the soft knowledge in hard form, and second, it helps in developing network infrastructure to facilitate the knowledge transfer across connected entities.

Culture reflects inculcated human attributes and behavior, for example, attitude toward sharing one's own experience to benefit the other team members of the organization. Another most important aspect of culture is the attitude toward learning. In recent times, one of the emerging emphasis of KM strategy is to promote organizational learning culture, so that it can easily adapt to the changing technological environment, and avoid the risk of process disruptions (Christensen, 1998). In one of the studies (Misra *et al.*, 2009), it has

been argued that training and learning is one of the success factors of ASD process. Learning culture is required not only to cope with the changes in the external technological environment, but also to increase the internal efficiency and sustainability of the organization. Organizations' internal work environment needs to preserve and sustain that knowledge that builds its core expertise and market value.

7.1 Supporting knowledge networks

A network can be viewed as a collection of homogenous of heterogenous nodes connected to each other through a number of direct or indirect links. The nodes can be considered as individuals, teams, and organizations. The nature of links may be formal or informal depending upon the nature of relational tie among those nodes. In an organization, there are formal networks of hierarchy, where authority and responsibility are well defined and the formal nature of information or knowledge flow is also well established. However, existence of some informal community networks among individuals are also identified in an organization, who frequently meet at the tea breaks, or during common gathering, and lot of knowledge exchange used to happen within them.

Nurturing informal networks can facilitate knowledge sharing in both co-located as well as in distributed agile teams that can contribute to knowledge creation. Developing support architectures to informal networks of communication can facilitate knowledge sharing in the organization without stymieing the normal information exchange. Such formal and informal networks when connected through the ICT (information and communication technology)-based infrastructure is known as knowledge networks (Seufert *et al.*, 1999). The nodes of the knowledge network are knowledge actors who participate in knowledge creation and sharing activities among themselves through the connecting links. The connecting links may be formal communication paths created by the structural hierarchy of the organization, or it may be informal ties with peer groups, friends, and relatives working in the same domain. The individuals constituting the knowledge network in our context of consideration are analysts, developers, testers, customers, and managers of the single agile team, of different agile teams, or those of traditional software development teams, all connected together through the same network of the organization, that is, practices both the agile and traditional methods for different projects.

In the context of KM, creating technological infrastructure for knowledge networks has been a central focus in order to facilitate knowledge transfer across the team and organization. To support knowledge sharing and learning in distributed agile teams, some such models have also been proposed (Shin, 2004) to form knowledge networks, but it does not specify the critical paths of knowledge exchange, in which one can focus and measure the frequency of knowledge exchange among the participants. Moreover, all kinds of communication and knowledge exchange are subjected to perform some specific tasks during the software development process through the assigned roles and responsibilities. These roles and responsibilities are well explained and tabulated in Section 3.

7.2 Exploring the critical paths of knowledge within the network nodes

Availability of a network-based infrastructure can strengthen the informal ties and facilitate the frequency of information flow across the network. This further improves the knowledge sharing culture within an organization. Identification of critical paths of network can assist to evaluate the nature and intensity of communication happening within and between the project development teams. This can provide an indirect assessment of learning potential and knowledge creation happening within the organization. Locating and focusing upon such specific paths can recommend specific measures to encourage the knowledge flow through those links that are critical to the sustainability of entire learning and sharing process. Once recognized, the frequency and direction of communication among the members can be further aligned according to the project goals. Here, we assumed a case where a software development organization who has adopted a mixed strategy, that is, used to apply both traditional as well as agile methods for different projects according to their nature, scope, and budget. A need arises to create a KM infrastructure to support learning, knowledge sharing, and hence sustainability of the agile practices. It should be developed in such a way so that it can be integrated with the KM infrastructure employed for traditional methods.

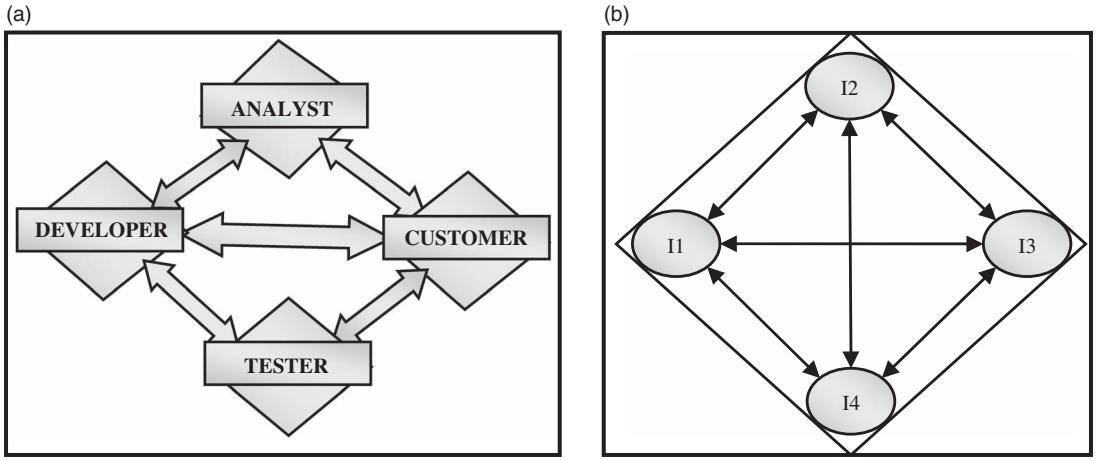

Figure 2 (a) Intra-team knowledge sharing paths for concurrent tasks in agile team. (b) Inter-team network of individuals between agile and non-agile teams

From the perspective of sustainability of organizational learning and knowledge flow, it is essential to identify the critical nodes of the organization through which knowledge flows and at which experience accumulates. These knowledge nodes are the individuals working together in the roles of customers, system analyst, developer, and tester on the key tasks of organization where knowledge is applied and DUI/JIT mode of learning takes place. Section 3 represents the functional classification of individuals under key roles, and key operational phases under various agile methods. These operational phases and assigned roles are similar to those in traditional software development process, but the degree of overlap, sequence of tasks, and repeatability of operations can dynamically change when considered under agile methods. For example, the process of requirements analysis and delivery scheduling can go in parallel. The key individuals involved during these overlapping tasks are system analysts and customers. Similarly, the processes of delivery scheduling and iteration plan can have some degree of overlap and are performed by the system analysts and software developers. One of the examples of concurrent operation maybe the case when system analyst, customers, and developers can engage and interact simultaneously to prepare system design and development planning, and settle the delivery schedule and iteration plans. A tester and customer jointly prepare the test plans and feedback reports for the further improvement of the product. A developer and tester can work jointly to incorporate the additional or corrected features into the product. At few moments, the developers, testers, and customers can be found working together to modify the old version of the software. The system analyst works in close proximity with the customer and the developer, while the tester works closely with the same individuals but have almost no interaction with the system analyst. All of such concurrent phases are accompanied by huge amount of knowledge exchange and knowledge creation. The network of critical communication paths for concurrent tasks of the ASD method can be represented as in Figure 2(a). The conceptual nodes of network are represented by blocks, which can represent an individual, or a group of them connected together. For example, the block named developer can be a network of developers of the same or different agile teams, or even of non-agile teams connected together through informal or digital networks. This network of same discipline (e.g. network of developers) is represented in Figure 2(b). This is a kind of inter-team network where there may be multiple individual having varying knowledge and experience levels. The knowledge flow happens from the more experienced individual to those having no or less experience. The individuals may belong to the same agile team, or different agile teams, or also to non-agile teams working on different projects of different parts of the same project. This kind of network can bridge the gap between knowledge repositories of the traditional software development environment to those of agile practices. The prime knowledge repositories of agile practices are individuals who stay connected to the knowledge repository of software development experts of traditional process and their explicit knowledge base.

7.3 Applying experience factory approach

The iterative process of agile methods requires that the knowledge created of previous iteration must be carried forward and given as input to the next iteration. More number of iterations requires more knowledge to be carried over in the mind and the knowledge should be retained till the least iteration is complete and development process is complete. This kind of knowledge creation and sharing happens predominantly through intensive verbal communication throughout the lifecycle of the project. In such kind of environment, there are some risky situations when any of the team members leave the project or becomes unavailable for some time. Then this will not only delay the project completion time, but may also disrupt the entire process. The agile methods are highly dependent upon the continuously busy experts, and have very less scope for grooming new experts to serve its future requirements. The other possible problem in knowledge transfer arises because of the closely packed schedule of the agile team experts. Despite having rich experience and the willingness to share it to make others learn from it, there may be scarcity of time, which is the most important resource for making any kind of knowledge transfer to happen.

To address such problems, one of the models suggested is to establish a separate support organization whose focus is to observe the main operations, collecting, and storing the experiential knowledge from the ongoing operations, structuring, and refining that knowledge, and giving it back to the mainstream operations of the project organization. The experience factory (Basili *et al.*, 1994) is a similar approach to help manage organizational experiences and support knowledge sharing. The experience factory approach can help the organization in multiple ways, as elicited by Basili *et al.* (2001):

- it can make the business less dependent upon the currently working employees;
- it can capture the experiences of experts and relieve them from the burden of carrying the knowledge;
- it has the potential of quickly imparting the skill and prepare new employees, which can be supplied to the main process whenever required;
- it improves the entire business process by analyzing, synthesizing, and refining the knowledge before supplying it back to the process.

The experience factory approach can help to manage even the dynamic possessions of knowledge (as described in Section 5) that can easily be captured by a separate observer organization, and which is difficult to be captured in any alternative mechanism. The experience factory is a closed loop process supporting learning and feedback between the project organization and observer organization. This can consider the software disciplines experimental, evolutionary, and non-repetitive characteristics and can be adapted to support the organizational learning environment (Basili *et al.*, 2001). This kind of organization has been successfully applied to the software organizations (Frank *et al.*, 1998). For our hypothetical organization, which uses both agile and traditional practices, this approach can be regarded as a very effective approach for KM. The observer (or support) organization who works as experience factory is formed by the people having work experience from the same domain and environment. For example, experienced analysts, system designer, and developers can become the members of the support organization and contributes to the management and maintenance of the experience factory. The KM practices of the organization can be synchronized in such a way that the same experience factory can serve to both agile and traditional teams. Even multiple teams can be linked to the same experience factory for its economically effective utilization.

The experience factory can capture both static as well as dynamic possession of knowledge in agile teams. We discussed earlier that dynamic possession of knowledge is hard to codify, because it does not stay with any individual but with the group. Therefore, a group observer is needed to capture the collectively owned dynamic state of knowledge. When it comes to knowledge supply by the support organization back to the project organization, there is comparatively more effort involved when it comes to serving the agile project team. Knowledge-based support to the traditional project teams are predominantly through hard form, while agile project team requires real-time knowledge-based support with direct interaction.

Figure 3 represents the integrated view of knowledge sharing and learning in the ASD team practices. In the figure, all the arrows denote the paths of knowledge flow between and within the network of agile

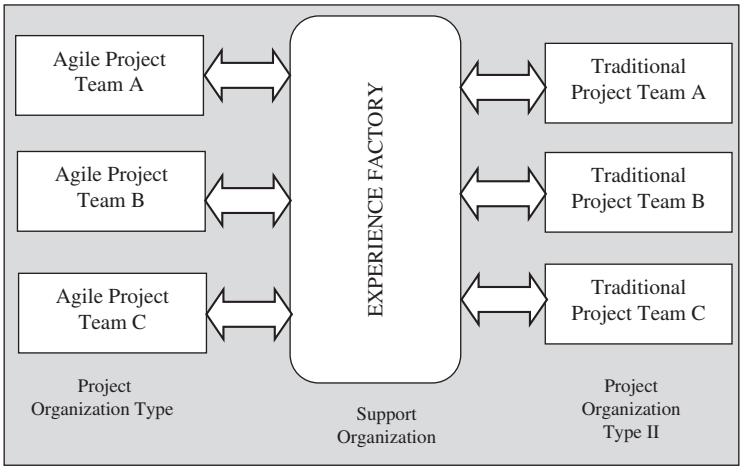

Figure 3 Graphical view of integrated approach

and non-agile project teams, and between software development teams and experience factory. Some of the critical network paths of communication has been discussed in previous section, and represented in Figures 2(a) and 2(b). A common knowledge repository is needed for accumulating the experiences and hard form of knowledge, and to capture the communication among the experts through the network. The knowledge recorded in the knowledge repository can be further refined and used by the experts of experience factory to provide support for the ASD team members.

7.4 Exploring the influence of nodal characteristics upon learning

Within the communication channels presented in Figures 2(a) and 2(b), knowledge transfer and learning happens between individuals having different functional and experiential characteristics. These can be seen as nodal characteristics of the knowledge network and influences the intensity of knowledge flow and learning. Following are the cases which should be considered seriously before developing the learning support system for ASD team:

1. Expert members of the same disciplines: this is the ideal and functionally most efficient group that can be employed in an agile team. Having highly experienced developers in a pair programming can be potentially effective, but it will not add much value to the knowledge creation or transfer to the new generation of developers and has nothing to add to the sustainability of the agile practices.
2. Expert members of different disciplines: this can make the information flow smoother and helps to create a quick understanding of the development environment. Experienced analysts and developers can take much lesser time to prepare the development plans based on the customers' requirements. Therefore, the saved time can be used for experience exchange with the other individuals of the inter-team network, and also has a good potential to support inter-team learning.
3. Expert and non-expert member of same functional discipline: this can be the ideal learning case to be focused upon and through which the organization can maintain the appropriate knowledge levels for prolonged periods. Implementation of this learning strategy can make the organization always equipped to mitigate the risk of knowledge loss happening because of employee migration. An agile team constitutes of multiple individuals working jointly in the same role to perform the assigned task. For example, in pair programming, two programmers can work on the same work station, one of them typing the code and other one reviewing the code. These two programmers frequently keep changing their roles and assisting each other. The variations in the skill level of these programmers can be introduced to allow learning during action. One of the members in the pair programming must be an expert to ensure the quality of the work and at the same time he can impart knowledge and perfection to the junior working with him.

4. Knowledge sharing between organizational and non-organizational entities: the ASD team has control only over their organizational entities not on the external factors. Knowledge creation and sharing is also a part of those controllable factors. However, customers' representatives also constitute the part of agile team who though not the part of organization, has a significant role to play in the knowledge exchange and influence the pace of the development process. Customers' representative has a crucial role: prepare the story cards, delivery schedules, iteration plan, and test feedback reports. An experienced customer always understands his products' functional requirements and can express it more specifically in terms of technical descriptors. This can help the organizational entities to understand, plan, and work exactly to what is demanded and finish the work right within the target deadlines. This can also save time for the ASD project team and attain customer satisfaction. However, in an alternative case, when the customers' representative is not having much experience and is unable to understand the technical terms, then it may take comparatively more time to communicate the functional requirements and development of test feedback report. The knowledge gap must be fulfilled by the agile team members to capture the customers voice quickly and transform it into their technical descriptors. Considering this situation, the ASD team can employ less-skilled members to work with the experienced and expert customers. This can provide space and scope for learning and knowledge creation to the organizational entities. However, inexperienced and less customers must be served by highly skilled and experienced ASD team to ensure the product, process, and service quality. In this situation, the organizational entities have to make additional effort to learn and understand.

8 Conclusions

Knowledge creation is a natural phenomenon during the software development process. To transfer the created knowledge from one individual to another, it requires additional time to share and learn, which is almost unavailable with the extremely busy agile project teams. On the other hand, to ensure the sustainability of the agile team and minimizing the adverse situations arising from exit of experienced members or entry of inexperienced members, knowledge sharing, and continuous learning are equally important. In this paper, first, we have developed the knowledge sharing and learning models for the agile teams. Then we have developed a KM framework for the sustainability of ASD practices, which can be integrated with the existing KM infrastructure working for the traditional projects, so that it can justify the scale and economic viability of the entire system. Our framework incorporates the application of knowledge networks and experience factory approach for integrating the KM practices.

Acknowledgements

Authors would like to thank Indian Institute of Kanpur for providing support and facilitations in carrying out this research.

References

Abrahamsson, P., Salo, O., Ronkainen, J. & Warsta, J. 2002. *Agile Software Development Methods: Review and Analysis*, VTT.

Agerfalk, P. & Fitzgerald, B. 2006. Flexible and distributed software processes: old petunias in new bowls? *Communication of the ACM* **49**(6), 26–34.

Alavi, M. & Leidner, D. E. 1999. Knowledge management systems: issues, challenges and benefits. *Communication of the Association for the Information Systems* **1**(7), 1–28.

Aurum, A. K., Daneshgar, F. & Ward, J. 2008. Investigating knowledge management practices in software development organisations? An Australian experience. *Information and Software Technology* **50**(6), 511–533.

Basili, B., Lindvall, M. & Costa, P. 2001. Implementing the experience factory concepts as a set of experience bases. *In Proceedings of 13th International Conference on Software Engineering & Knowledge Engineering*.

Basili, V. R., Caldiera, G. & Rombach, A. D. 1994. The experience factory. *Encyclopedia of Software Engineering -2*, Volume Set, John Wiley & Sons, Inc., 469–476.

Bjornson, F. O. & Dingsoyr, T. 2008. Knowledge management in software engineering: a systematic review of studied concepts, findings and research methods used. *Information and Software Technology* **50**(11), 1055–1068.

Bohn, R. E. 1997. Measuring and managing technological knowledge. *IEEE Engineering Management Review*, 77–88.

Brandenburg, D. C. & Ellinger, A. D. 2003. The future: just-in-time learning expectations and potential implications for human resource development. *Advances in Developing Human Resources* **5**(3), 308–320.

Buono, A. F. & Poulfelt, F. 2005. *Challenges and Issues in Knowledge Management*, Information Age Publishing.

Carter, C. & Scarbrough, H. 2001. Toward a second generation of KM? The people management challenge. *Education + Training* **43**(4/5), 215–224.

Chau, T. & Maurer, F. 2004. Knowledge sharing in agile software teams. In *Logic Versus Approximation*, W. Lenski (ed.), Springer-Verlag, 173–183.

Christensen, C. 1998. *The Innovator's Dilemma: When New Technologies Cause Great Firms to Fail*, Harvard Business School Press.

Cockburn, A. 2000. Agile software development draft version: 3b. `http://www.snip.gob.ni/xdc/Agile/AgileSoftwareDevelopment.pdf` (accessed 17 August 2009).

Corbin, R. D., Dunbar, C. B. & Zhu, Q. 2007. A three-tier knowledge management scheme for software engineering support and innovation. *Journal of Systems and Software* **80**(4), 1494–1505.

Frank, H., Kurt, S. & Eva, W. 1998. Establishing experience factories at Daimler-Benz an experience report. In *Software Engineering, 1998. Proceedings of the 1998 International Conference on*, pp. 443–447. IEEE, 1998.

Guo, A., Yubing, H., Chen, J. & Tao, T. 2008. Matching the STI/DUI mode of learning dynamically to realize continuous innovation: a case study of CIMC Group. In *4th International Conference on Management of Innovation and Technology, IEEE Xplore*, 794–797.

Hildreth, P. M. & Kimble, C. 2002. The duality of knowledge. *Information Research* **8**, 1.

Jorgensen, M. & Moløkken-Østvold, K. 2006. How large are the software cost overruns? A review of the 1994 CHAOS report. *Information and Software Technology* **48**(4), 297–301.

Lundvall, B. A. & Lorenz, E. 2007. Modes of innovations and knowledge taxonomies in the learning economies. In *CAS Workshop on Innovation in Firms*. `http://www.cas.uio.no/research/0708innovation/CASworkshop_LundvallLorenz.pdf`

Misra, S. C., Kumar, V. & Kumar, U. 2009. Identifying some important success factors in adopting agile software development practices. *The Journal of Systems and Software* **82**(11), 1869–1890.

Modesitt, K. L., Maxim, B. R. & Akingbehin, K. 1999. Just in time learning in software engineering. *JI. of Computers in Mathematics and Science Teaching* **18**(3), 287–301.

Nonaka, I. & Konno, N. 1998. The concept of ba: building foundation for knowledge creation. *California Management Review* **40**(3), 40–54.

Novak, J. & Wurst, M. 2005. Supporting knowledge creation and sharing in community based on mapping implicit knowledge. *Journal of Universal Computer Science* **10**(3), 235–251.

Probst, G., Buchel, B. & Raub, S. 1999. Knowledge as a strategic resource. In *Knowledge in Firms, Understanding, Managing and Measuring Knowledge*, Krogh von, G., Roos, J. & Kliene, D. (eds). Sage, 240–250.

Quintas, P., Lefrere, P. & Jones, G. 1997. Knowledge management: a strategic agenda. *Long Range Planning* **30**(3), 385–391.

Qumer, A. & Henderson-Sellers, B. 2006. Comparative evaluation of XP and scrum using the 4D analytical tool (4-Dat). In *European and Mediterranean Conference on Information Systems (EMCIS).*

Qumer, A. & Henderson-Sellers, B. 2008a. A framework to support the evaluation, adoption and improvement of agile methods in practice. *The Journal of Systems and Software* **81**(11), 1899–1919.

Qumer, A. & Henderson-Sellers, B. 2008b. 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.

Qumer, A., Henderson-Sellers, B. & McBride, T. 2007. Agile adoption and improvement model. In *Proceedings of European and Mediterranean Conference on Information Systems*.

Redecker, C. 2009. Review of learning 2.0 practices: study in the impact of Web 2.0 innovations on education and training in Europe. JRC Scientific and Technical Reports, European Commission.

Seufert, A., Krogh, G. v., Bach, A. 1999. Towards knowledge networking. *Journal of Knowledge Management* **3**(3), 180–190.

Shin, M. 2004. A framework for evaluating economics of knowledge management systems. *Information and Management* **42**(1), 179–196.

Sullivan, P. H. 1999. Profiting from intellectual capital. *Journal of Knowledge Management* **3**(2), 132–142.

Turk, D., France, R. & Rumpe, B. 2002. Limitations of agile software processes. `http://www.agilealliance.org/show/1096` (accessed 17 August 2009).

Wenger, E. 1996. Communities of practice: the social fabric of a learning organization. *Healthcare Forum Journal* **39**(4), 20–26.

Wissensmanagement Forum 2003. An illustrated guide to knowledge management. `www.wm-forum.org` (accessed 10 June 2008).