Creative Thinking in eXtreme Programming

Broderick Crawford[1] ,
Claudio León de la Barra[2],
Ricardo Soto [3],
Sanjay Misra[4],
Eric Monfroy[5]

[1] Pontificia Universidad Católica de Valparaíso, Valparaíso, Chile
[2] Universidad Finis Terrae, Santiago, Chile
[3]Universidad Autónoma de Chile, Santiago, Chile
[4]Covenant University, Nigeria
[5]LINA, Université de Nantes, France
Contact(s). Broderick.Crawford@ucv.cl, Claudio.LeondelaBarra@ucv.cl, Ricardo.Soto@ucv.cl, Sanjay.Misra@covenantuniversity.edu.ng, Eric.Monfroy@univ-nantes.fr

***Abstract*:** Agile methods such as eXtreme Programming have achieved an explosive interest in the software development community. They can be seen as a reaction to the more traditional and control-oriented methods, agile methods handle changes in design and requirements and they open up for creativity during the whole project lifecycle. The knowledge management in agile methods is also agile, it means that knowledge creation and sharing processes are simplified in comparison with other more comprehensive development methodologies. This paper is developed under the idea that agile software development can be enhanced by a better understanding of knowledge management and creativity. eXtreme Programming is analyzed from the perspective of the creativity, we believe that concepts related to creative teams (roles, structure, performance and purposes) are important insights about the use of agile methods in general and eXtreme Programming in particular.

Keywords/Index Terms: Knowledge Management; Creativity; Software Engineering; Agile Methods; User-centered innovation.

1. Introduction

In a globalized and knowledge based economy, firms continuously need to increase efficiency and to innovate in order to achieve a competitive advantage and to survive (Veryzer, 1998). New product developers have recognized that they need to inject more customer know-how into their product innovation processes, encouraging the direct interaction of development team with customers, in contrast with traditional practices. The integration of customer know-how into the development of new products leads to a higher degree of

innovation, reduced risks and more precise resource speeding (Gassmann et al., 2005).

Nowadays, the paradigm shift from producer-centered to user-centered innovation describes the shift from the concepts of innovation activities considered the domain of specialist producers to a notion that embraces the active role of the end user in the innovation process. Numerous initiatives by firms like Dell, Procter & Gamble and Starbucks for integrating user-generated innovations into the firms indicate the usefulness and importance of this approach. Research in the field of new product development and innovation management suggests that effective product development requires interplay between developers and customers (von Hippel et al., 2001). In software engineering it is the same (Sandmeier, 2009). Such as the insights from existing research, the eXtreme Programing method from software engineering and the successful practices have enabled to derive a model of extreme innovation (Sandmeier & Gassmann, 2006; Gassmann et al., 2006). Extreme innovation allows from customer feedback and ideas to be integrated directly into new product development. The innovation process is iterative, taking a step approach to innovation, where each new phase of development is tested with customers. It requires a flexible approach to project management and a corresponding structural organization.

Room for creativity is a key in software development today; you need creativity for building software (Gutbrod & Wiele, 2012). A fruitful way to think about software development is to consider it as a cooperative game of invention and communication (Cockburn, 2006).

Until the acceptance of programming in pairs, an agile practice in which two people sit together and co-write programs (Beck, 2000), the programmers accented the invention portion of the cooperative game. Today, communication is equally important. The innovation trends emphasize firms ability to draw from external knowledge and effectively diffuse and use knowledge within the firms as well as outside the firms, e.g., in co-action processes with stakeholders across a multitude of disciplines for the innovation success. In order to develop quality software, teams need to leverage the knowledge of each team member skills (Neves et al., 2011; Santos et al., 2013; Santos & Goldman, 2011).

Software engineering is a

knowledge intensive discipline where its activities require the use and sharing of knowledge between the stakeholders. Then, a better transfer and application of the knowledge aim to foster the software processes, whether these are done using traditional or agile approaches (Patel et al., 2012).

In software organizations the knowledge held by the employees is the main asset and software development projects depend mostly on team performance: “software is developed for people and by people" (John et al., 2005). But surprisingly, most of software engineering research is technical and deemphasizes the human and social aspects. By other hand, the traditional development process of new products that is a fundamental part in the marketing has been recently criticized by Kotler and Trías de Bes (Kotler & Trías de Bes, 2004). They point out that fundamental creative aspects are not considered at all and as a consequence this development is not useful, viable or innovative. In this context, it is interesting to consider the new proposals of agile methodologies for software development in order to analyse and evaluate them at the light of the existing creative expositions, mainly considering the teamwork practices.

The agile principles and values emphasize the collaboration and the interaction in software development and the creative work, by other side, involves collaboration in some form and it can be understood as an interaction between an individual and a sociocultural context (Sanz & Misra, 2011).

We believe that the innovation and development of new and usable products is an interdisciplinary issue (Takeuchi & Nonaka, 1986; Nonaka & Takeuchi, 1995). We are interested in the study of the potential of new concepts and techniques to foster knowledge management and creativity in agile software development in general and eXtreme Programming in particular (Gu & Tong, 2004; Crawford et al., 2012; Crawford et al., 2008).

This paper is organised as follows: Section 2 presents Agile Methods, Section 3 is dedicated to the presentation of Knowledge Management, Section 4 presents the background and general concepts on Creativity and in Section 5 we conclude the paper.

2. Agile Software Methods

Agile methods are based on iterative and incremental development, where requirements and its software solutions evolve through collaboration between

cross functional teams. They promote flexible planning, evolutional development and delivery, an iterative approach, and encourages fast response to change. It is a conceptual proposal introduced in the Agile Manifesto in 2001 (Beck, 2001). Through this declaration its adherents have come to value: "individuals and interactions over processes and tools, working software over comprehensive documentation, customer collaboration over contract negotiation and responding to change over following a plan".

These new methods attempt a useful balance between no process and too much process, providing just enough process to gain a reasonable reward, resulting that agile methods have significant differences with the previous engineering methods (Fowler, 2001):

Agile methods are adaptive rather than predictive. Engineering methods tend to try to plan out a large part of the software process in great detail for a long span of time, this works well until things change. So, their nature is to resist change. Agile methods, however, welcome change. They are processes that try to adapt and thrive on change, even to the point of changing themselves.

Agile methods are people oriented rather than process oriented. The goal of engineering methods is to define a process that will work well whoever happens to be using it. Agile methods assert that no process will ever make up the skill of the development team, so the role of a process is to support the development team in their work. Scrum and eXtreme Programming are the most used agile software development methods.

2.1 Scrum

Scrum adapts aspects from complexity theory, systems dynamics and theory of knowledge creation setting a project management agile software approach (Moe et al., 2010). A relevant characteristic in Scrum is the self-management, representing a new method for to plan and to manage projects. It provides team members the chance for mutual recognition of competences. It is a straight vehicle for communication, collaboration, trust and cohesion. The term Scrum was first adapted as a metaphor, from (Nonaka & Takeuchi, 1995) referring to the holistic action of an entire rugby team going the entire distance, together.

In general, Scrum is described as a development process for small teams which includes a series of

short development iterations named "sprints". Scrum is an iterative, incremental framework (Schwaber & Beedle, 2001), the sprints are typically 1-4 weeks in length, and which take place one after the other, they are of fixed duration (they end on a specific date whether the work has been completed or not) and are never extended. At the beginning of each sprint, a cross-functional team prioritizes items from a list of requirements, and commits to complete them by the end of the sprint; during the sprint, the deliverable does not change. Each work day, the team gathers briefly to report to each other on progress, and update simple charts that orient them to the work remaining. At the end of the sprint, the team demonstrates what they have built, and gets feedback which can then be incorporated in the next sprint.

2.2 eXtreme Programming XP

Extreme Programming is based on values of simplicity, communication, feedback, and courage. It works by bringing the whole team together in the existence of simple practices, with enough feedback to enable the team to see what they are doing and where they are. In XP, every member of the project is an integral part of the whole team and plays a specific role (Beck, 2000).

Roles in XP

XP defines the following roles for a software development process (Beck, 2000).

The customer defines what to do (user stories) and in what order (planning game), in XP the customer is also responsible for the requirements because the stories are written by him. Additionally, functional testing are derived and verified by him from the stories with the help from the Tester.

The programmer, it is a very important role because XP is a programmer-centric methodology. It does not make use of specialists like analysts, software architects or software designers. Instead this work is performed by the programmers. The programmer must have different skills, mainly: communication (XP relies on face-to-face communication), coding and the ability to work in teams (especially with collective code ownership and programming in pairs).

The managing part of an XP project is divided into two roles: the coach and the tracker. The coach is responsible for the technical execution of the project. The job of the tracker is to gather whatever metrics are being tracked for the project (tracking is not really full time, it is usually performed by the coach or a programmer).

The tester, in contrast to other roles, he has only few responsibilities. This is due to the fact that most of the white box testing is performed by the programmers. The tester helps the customer to write the functional tests and run them when the tests that cannot be automated (the role of tester is not filled by a dedicated person but by one of the programmers or the tracker).

The consultant will be hired when the project needs deeper (technical) knowledge. The consultant is hired to provide knowledge. The consultant transfers this knowledge to the team members, enabling them to solve the problem on their own. The Big Boss or Manager provides the resources for the process. The big boss needs to have the general picture of the project, be familiar with the current project state, and know whether any interventions are needed to ensure the success of the project.

3. Knowledge Management

One of the most widely accepted approaches to classifying knowledge from a KM perspective is the Knowledge Matrix of Nonaka and Takeuchi (Nonaka & Takeuchi, 1995). This matrix classifies knowledge as either explicit or tacit, and either individual or collective. Nonaka and Takeuchi also proposes corresponding knowledge processes that transform knowledge from one form to another: socialization (from tacit to tacit, whereby an individual acquires tacit knowledge directly from others through shared experience, observation, imitation and so on); externalization (from tacit to explicit, through articulation of tacit knowledge into explicit concepts); combination (from explicit to explicit, through a systematization of concepts drawing on different bodies of explicit knowledge); and internalization (from explicit to tacit, through a process of learning by doing and through a verbalization and documentation of experiences). Nonaka and Takeuchi model the process of organizational knowledge creation as a spiral in which knowledge is amplified through these four modes of knowledge conversion.

3.1 Knowledge Management in Software Engineering

The main argument to Knowledge Management in Software Engineering is that it is a knowledge intensive activity. Software development is a process where every person involved has to make a large number of decisions and individual knowledge has to be shared and leveraged at a project and organization level, and this is exactly what KM proposes. People

in such groups must collaborate, communicate, and coordinate their work, which makes knowledge management a necessity.

In software development one can identify two types of knowledge: Knowledge embedded in the products or artifacts, since they are the result of highly creative activities and Meta-knowledge that is knowledge about the products and processes (Rus & Lindvall, 2002; Mentzas, 2000; Apostolou & Mentzas, 2003).

4. Creativity

Creativity is defined as the capacity to generate or recognize original, elaborated and useful ideas (Amabile, 1996). By self the creative is an act of knowledge creation (Sung & Choi, 2012). Although the creativity can be approached from the individual's perspective, its greatest potential and development is appreciated at team level (Amabile, 1998; Leenders et al., 2003; Gilson & Shalley, 2004; Chen, 2006).

4.1 Creativity in Software Development

Software engineering is a knowledge intensive process that includes some aspects of Knowledge Management and Creativity in all phases: eliciting requirements, design, construction, testing, implementation, maintenance, and project management (John et al., 2005). No worker of a development project has all the knowledge required to fulfill all activities. This underlies the need for communication, collaboration and knowledge sharing support to share domain expertise between the customer and the development team (Chau et al., 2003).

The plan-driven approaches, like the waterfall model and its variances, facilitate knowledge sharing through documentation. They also promote usage of roles with functional specialization in the teams and detailed plans of the entire software development project. It shifts the focus from individuals and their creative abilities to the processes themselves. By other hand, agile methods emphasize and value individuals and interactions over processes. Plan-driven or tayloristic methods heavily and rigorously use documentation for capturing knowledge gained in the different activities of the life-cycle (Chau & Maurer, 2004). In contrast, agile methods suggest that most of the written documentation can be replaced by enhanced informal communication among team members internally and between the team and the customers. Thereby, the agile way is with a stronger emphasis on tacit knowledge rather than explicit

knowledge (Beck et al., 2001).

A way to improve software development methods is to design a process which can encourage the creativity of the developers. There are few studies reported on the importance of creativity in software development. In management and business, researchers have researched about creativity evidencing that the workers who had appropriate creativity characteristics, worked on complex, challenging jobs, and were supervised in a no controlling fashion produced more creative work. Accordingly, the use of creativity in software development is undeniable, but requirements engineering is not recognized as a creative process in all the cases (Maiden et al., 2004). In a few publications the importance of creativity has been investigated in all the phases of software development process (Gu & Tong, 2004; Glass, 1995; Crawford & León de la Barra, 2007; León de la Barra & Crawford, 2007; Crawford et al.,2008; Crawford & León de la Barra, 2008) and mostly focused in the requirements engineering (Robertson, 2005; Mich et al., 2005; Nguyen & Cybulski, 2008; Nguyen & Shanks, 2009). Nevertheless, the use of techniques to foster creativity in requirements engineering is still shortly investigated. It is not surprising that the role of communication and interaction is central in many of the creativity techniques. The most popular creativity technique used for requirements identification is the classical brainstorming and more recently, role-playing-based scenarios, user stories, storyboard-illustrated scenarios, simulating and visualizing have been applied as an attempt to bring more creativity to requirements elicitation. These techniques try to address the problem of identifying the viewpoints of all the stakeholders (Mich et al., 2005; O'hEocha & Conboy, 2010).

However, in requirements engineering answers are not evident, it is indispensable to ask, observe, discover, and increasingly create requirements. If the goal is to build new and innovative products, we must make creativity part of the requirements activities. Indeed, the importance of creative thinking is expected to increase over the next decade (Maiden & Gizikis, 2001). In (Robertson, 2005; Robertson, 2002) very interesting open questions are proposed: is inventing part of the requirements activity? It is if we want to advance. So, who does the inventing? Requirements analysts are ideally placed to innovate. They understand the business problem, have updated knowledge of the technology, will be blamed if the

new product does not please the customer, and know if inventions are appropriate to the work being studied. We cannot rely on the customer to know what to invent. The designer sees his task as deriving the optimal solution to the stated requirements. We cannot rely on programmers because they are far away from the work of client to understand what needs to be invented.

In short, requirements analysts are the people whose skills and position allows, indeed encourages, creativity.

In (Boden, 2004) the author, a leading authority on cognitive creativity, identifies basic types of creative processes: exploratory creativity explores a possible solution space and discovers new ideas, combinatorial creativity combines two or more ideas that already exist to create new ideas, and transformational creativity changes the solution space to make impossible things possible. Then, most requirements engineering activities are exploratory, acquiring and discovering requirements and knowledge about the problem domain. But, requirements engineering practitioners have explicitly focused on combinatorial and transformational creativity.

4.2. Creative Process

The creative process is the main aspect of team performance, because it supposes a series of clearly defined phases to be realized by the team members in order to obtain a concrete creative result. The phases, considering the ideas of Wallas (Wallas, 1926) and Leonard and Swap (Leonard & Swap, 1999), are the following ones:

1) *Initial preparation*: the creativity will bloom when the mental ground is deep, fertile and it has a suitable preparation. Thus, the deep and relevant knowledge, and the experience precedes the creative expression.

2) *Encounter*: the findings corresponding to the perception of a problematic situation.

3) *Final preparation*: it corresponds to the understanding and foundation of the problem. It is the immersion in the problem and the use of knowledge and analytical abilities. It includes search of information and the analysis of variables.

4) *Generation of options*: referred to produce a set of alternatives. It supposes the divergent thinking. It includes, on one hand, finding principles, lines or addresses, when making associations and uniting different references and, on the other hand, to generate possible solutions, combinations and interpretations.

5) *Incubation*: it corresponds to

the required time to reflect about the elaborated alternatives, and to "test them mentally".

6) *Options Choice*: it corresponds to the final evaluation and selection of the options. It supposes the convergent thinking.

7) *Persuasion*: closing of the creative process and communication to other persons.

Because it is not a linear process, for each one of the defined phases it is possible to associate feedbacks whose "destiny" can be anyone of the previous phases in the mentioned sequence.

The team performance is directly determined by the creative process (Kotler & Armstrong, 2003; Leonard & Swap, 1999). It is interesting to interrelate the phases of XP with the phases considered in a creative process.

The initial preparation defined in the creative process corresponds to the exploration phase in XP, where the functionality of the prototype and familiarization with the methodology are established.

The final stage of preparation is equivalent with the phases of exploration and planning in XP, defining more in detail the scope and limit of the development.

The option generation phases, incubation and election of options defined in the creative process correspond to the iterations made in XP and also with the liberations of the production phase (small releases). In XP there is not a clear distinction of the mentioned creative phases, assuming that they occur to the interior of the team.

The feedback phase (understanding this one as a final stage of the process, and not excluding that can have existed previous micro-feedbacks since the creative process is nonlinear) it could correspond in XP with the maintenance phase.

The persuasion phase is related to the phase of death established in XP, constituting the close of the development project with the final liberation.

4.3. Roles in a Creative Team

Lumsdaine and Lumsdaine (Lumsdaine & Lumsdaine, 1995) proposed the cognitive abilities required to creative problem resolution, the different roles considered are the following ones:

1) The Detective is in charge of collecting the greatest quantity of information related to the problem.

2) The Explorer detects what can happen in the area of the problem and its context. He thinks on its long term effects and he anticipates certain situations that can affect the context. The explorer perceives the problem in a broad sense.

3) The Artist creates new things,

transforming the information. He must be able to break its own schemes to generate eccentric ideas, with imagination and feeling.

4) The Engineer is the one in charge of evaluating new ideas. He makes the idea convergence process, in order to clarify the concepts and to obtain practical ideas that can be implemented for the resolution of problems.

5) The Judge must do a hierarchy of ideas and decide which of them will be implemented. Additionally, he must discover faults or inconsistences in the solutions.

6) The Producer is in charge of the implementation of the chosen ideas.

Leonard and Swap (Leonard & Swap, 1999) have mentioned possible additional roles, trying to improve the divergence and the convergence in the process:

The provoker takes the members of the team "to break" habitual mental and procedural schemes to allow the mentioned divergence (in the case of the "artist") or even a better convergence (in the case of the "engineer").

Think tank that it is invited to the team sessions to give a renewed vision of the problem-situation based on his expertise and experience.

The facilitator helps and supports the team work in its creative task in different stages.

The manager cares for the performance and especially for the results of the creative team trying to adjust them to the criteria and rules of the organization (use of resources, due dates).

Kelley and Littman (Kelley and Littman, 2005), on the other hand, have proposed a role typology similar to Lumsdaine and Lumsdaine (Lumsdaine & Lumsdaine, 1995), being interesting that they group the roles in three categories: those directed to the learning of the creative team (corresponding with the detective, explorer, artist, provoker and think tank), others directed to the internal organization and success of the team (similar to the judge, facilitator and manager) and roles whose purpose is to construct the innovation (related to the role of the engineer and judge).

The following is the correlation between creative and XP roles:

The client in XP plays the role of detective, collecting the information related with the problem, he generates the first contact with the software development team.

The function of explorer consisting in defining completely the problem is made in XP as much by the client as the manager of the team, all

together appreciate the reach of the identified problem, as well as of the possible solutions.

The developer that in XP methodology is in charge of the analysis, design and programming of software does the function of the artist, consisting in transforming the information, creating new relations, and therefore generating interesting solutions.

The function of the engineer referred to clarify and to evaluate the new ideas, in terms of its feasibility is made in XP by the tester and the tracker.

In XP the tracker and the client play the function of the judge, understood as the definitive selection of the solutions to implant.

In XP the client (in his organization) plays the role of the producer, referred to the implementation of the selected ideas (strictly speaking it is working software), including the processes and procedures that this function implies.

The supporting roles considered are:

Creativity demands that the divergence as well as convergence in the solutions to be maximum and complete. There is not explicit reference in XP methodology about divergent thinking. It is interesting to consider the provoker.

The XP role of the consultant is equivalent to the think tank in creativity, helping the team to work "from outside".

The coach in XP corresponds to the facilitator whose function is helping the team.

The manager whose function is to lead to the team in terms of its general efficiency and its effectiveness corresponds with big boss or manager in XP.

4.4. Basic Organizational Conditions

Respecting to the structural dimension of a new product development team, it is possible to relate the roles in creativity to the roles defined in the agile methodology separating base roles (those directly related to the creative processes and software development) and support roles (whose function is to support or lead the other roles for a better performance). Furthermore, it is important to considerate how the team can operate. In order to implement the functionality of each role, we must considerate two aspects: basic organizational conditions and the pertinent creative process.

The creative team performance is determined by the organizational conditions in which it is inserted

(Amabile, 1998; Isaksen et al., 1999; Leonard & Swap, 1999). Some conditions are necessary, although not sufficient, for the creative performance: autonomy, communication, cooperation, learning, handling of conflicts, pressure, the formalization, evaluation of performance, resources availability and atmosphere of work.

The autonomy refers to the capacity of the people and the whole team to act and make decisions. By example, this aspect is related to the following XP practices: the client in situ, since it is part of the team and, in addition, has decisional capacity delegated by its own organization; the use of metaphors, of codification standards and the existence of "right" rules really represent codes of shared thought and action, that make possible the autonomy of the team members; the small deliveries and the fact of the collective property allow that all the involved ones share official and explicit knowledge, that results in a greater independence of the members and the possibility of a minor coordination among them.

The communication, cooperation and learning of members are fortified since the client is present and there exist opened spaces to work together and in a pair programming mode. The dynamic of work is based on planning game and metaphors involving all the participants from the beginning (client and equipment developer). Also, the use of codification standards, the small deliveries, the collective property of the code and the simple design, allow that the person has clear performance codes and rules about what is expected and acceptable (internal culture) in order to establish the required communication and cooperation.

The handling of possible conflicts between the client and the development team, and internally at team level is favored by XP practices handling it (presence of the client, pairs programming, planning game, continuous integration, tests, collective property), or to reduce it and to avoid it (small deliveries, simple design, forty hour a week and codification standard).

In creativity the pressure is appraised as favorable until certain degree, it is present in XP through the client in situ, the programming by pairs, the planning game, the tests and continuous integration. It is possible to avoid, or at least to reduce, the pressure through the re-factorization, the small deliveries, the collective property, and the fact that surpassing the forty weekly working hours is seen like an error.

The formalization gives account of all those formal aspects defined explicitly and that must be known and shared by the whole team. It is assured in XP through planning game, metaphors, continuous integration, the collective property, the forty hours per week and the codification standards guiding the desirable conduct and performance of the team.

The evaluation of the performance is made in XP through pair programming (self-evaluation and pair evaluation), frequent tests and even through the forty weekly hours (a metric indicating the limit of effectiveness), all at the light of the planning. Finally, the presence of client constitutes the permanent and fundamental performance evaluation of the team and the software products. These evaluation characteristics empower the learning process.

The time dedicated has fundamental importance in XP team respecting the available resources. This aspect is strongly stressed in creativity. The pair programming and the developer multifunctional role allow optimizing the partial working-times, as well as the whole project time, ensuring a positive pressure.

The atmosphere of work, referred in creativity to the surroundings that favor or make difficult the creative performance (including aspects like available spaces, noise, colors, ventilation, relaxation places …) are assured only partially in XP through the open spaces, as a way to assure the interaction between members of the team.

5. Conclusion

This paper is a call for creativity enhancing agile software development. It has presented some approaches for improving the XP team structure and operation. Meanwhile, we are taking a walk through two questions:

How KM practices should be integrated with agile software?

In Software Engineering many development approaches work repeating the basic linear model iteratively. Then, in a lot of cases an iterative development approach is used to provide rapid feedback and continuous learning in the development team. To facilitate learning among developers, agile methods use daily or weekly stand up meetings, pair programming and collective ownership. Agile methods emphasize on people, communities of practice, communication, and collaboration in facilitating the practice of sharing tacit knowledge at a team level. An important finding is the need to not focus exclusively on explicit knowledge but also on tacit

knowledge.

They also foster a team culture of knowledge sharing, mutual trust and care. Agile development is not defined by a small set of practices and techniques. Agile development defines a strategic capability, a capability to create and respond to change, a capability to balance flexibility and structure, a capability to draw creativity and innovation out of a development team, and a capability to lead organizations through turbulence and uncertainty. They rough out blueprints (models), but they concentrate on creating working software. They focus on individuals and their skills and on the intense interaction of development team members among themselves and with customers and management.

How creativity can enhance agile practices?

By other side, Creativity and innovation are essential skills in almost any teamwork. Having a team that can solve problems quickly and effectively with a little creative thinking is beneficial to everyone. The performance of a team depends not only on the competence of the team itself in doing its work, but also on the organizational context. The organizational conditions in which the team is inserted are very important too. If workers see that their ideas are encouraged and accepted, they will be more likely to be creative, leading to potential innovation in the workplace. The creation of a collaborative work environment will foster the communication between employees and reward those that work together to solve problems. Encouraging team members to take risks, the opposite of creativity is fear. Then, it is necessary to create an environment that is free from fear of failure: failures are a learning tool.

We believe that knowledge management and creativity enhancing agile software development can be aligned with the design of high quality software. Here, we provided an understanding of knowledge management and creativity in relation with new software engineering trends.

References

Amabile, T. (1996). Creativity in Context: Update to the Social Psychology of Creativity. Westview Press.

Amabile, T. (1998). How to kill creativity. Harvard Business Review, Sept-Oct:77-87.

Apostolou, D. & Mentzas, G. (2003). Experiences from knowledge management implementations in companies of the software sector.

Business Process Management Journal, 9(3).

Beck, K. (2000). Extreme programming explained: embrace change. Addison-Wesley Longman Publishing Co., USA.

Beck, K. (2001). Agile alliance. http://agilemanifesto.org.

Beck, K., Beedle, M., Bennekum, A. V., Cockburn, A., Cunningham, W., Fowler, M., Grenning, J., Highsmith, J., Hunt, A., Jeries, R., Kern, J., Marick, B., Martin, R. C., Mellor, S., Schwaber, K., Sutherland, J., & Thomas, D. (2001). Manifesto for agile software development. Available at http://agilemanifesto.org.

Boden, M. (2004). The Creative Mind: Myths and Mechanisms. Routledge, USA.

Chau, T. & Maurer, F. (2004). Knowledge sharing in agile software teams. In Lenski, W., editor, Logic versus Approximation: Essays Dedicated to Michael M. Richter on the Occasion of his 65th Birthday, volume 3075 of Lecture Notes in Articial Intelligence, pages 173-183. Springer.

Chau, T., Maurer, F., & Melnik, G. (2003). Knowledge sharing: Agile methods versus Tayloristic methods. Twelfth International Workshop on Enabling Technologies: Infrastructure for Collaborative Enterprises, WETICE, pages 302-307.

Chen, M. H. (2006). Understanding the benefits and detriments of conflict on team creativity process. Creativity and Innovation Management, 15(1):105-116.

Cockburn, A. (2006). Agile Software Development: The Cooperative Game (2nd Edition) (Agile Software Development Series). Addison-Wesley Professional.

Crawford, B. & León de la Barra, C. (2007). Enhancing creativity in agile software teams. Lecture Notes in Computer Science,4536:161-162.

Crawford, B. & León de la Barra, C. (2008). Integrating creativity into extreme programming process. In Cordeiro, J. and Filipe, J., editors, ICEIS, pages 216-219.

Crawford, B., León de la Barra, C., & Letelier, P. (2008a). Communication and creative thinking in agile software development. In Cascini, G., editors, Computer-Aided Innovation (CAI), volume 277 of The International Federation for Information Processing, pages 205-216. Springer US.

Crawford, B., León de la Barra, C., & Rubio, J. (2008b). Knowledge sharing in traditional and agile software processes. In Cordeiro, J.,Shishkov, B., Ranchordas, A., and Helfert, M., editors, ICSOFT (PL/DPS/KE), pages 376-379. INSTICC Press.

Crawford, B., León de la Barra, C., Soto, R., Misra, S., & Monfroy, E. (2012). Knowledge management and creativity practices in software engineering. In Liu, K. and Filipe, J., editors,KMIS, pages 277-280. SciTePress.

Fowler, M. (2001). The new methodology. Available at www.martinfowler.com/articles/newMethodology.html

Gassmann, O., Sandmeier, P., & Wecht, C. (2006). Extreme customer innovation in the front-end: learning from a new software paradigm. IJTM, 33(1):46-66.

Gassmann, O., Wecht, C., & Sandmeier, P. (2005). Early customer integration eci new trends and developments beyond the lead user ap-proach. In European Academy of Management Conference (EURAM), Munich.

Gilson, L. L. & Shalley, C. E. (2004). A little creativity goes a long way: An examination of teams engagement in creative processes. Journal of Management, 30(4):453-470.

Glass, R. (1995). Software creativity. Prentice-Hall, USA.

Gu, M. & Tong, X. (2004). Towards hypotheses on creativity in software development. PROFES, 3009:47-61.

Gutbrod, R. & Wiele, C. (2012). The Software Dilemma: Balancing Creativity and Control on the Path to Sustainable Software. Management for professionals. Springer Berlin Heidelberg.

Isaksen, S., Lauer, K., & Ekvall, G. (1999). Situational outlook questionnaire: A measure of the climate for creativity and change. Psychological Reports, pages 665-674.

John, M., Maurer, F., & Tessem, B. (2005). Human and social factors of software engineering: workshop summary. ACM SIGSOFT Softw. Eng., Notes, 30:1-6.

Kelley, T. & Littman, J. (2005). The Ten Faces of Innovation: IDEOs Strategies for Defeating the Devil's Advocate and Driving Creativity. Throughout Your Organization. Doubleday Random House, USA.

Kotler, P. & Armstrong, G. (2003). Principles of Marketing. Prentice Hall, New Jersey.

Kotler, P. & Trías de Bes, F. (2004). Marketing Lateral. Editorial Pearson/Prentice Hall, Spain.

Leenders, R. T., van Engelen, J. M., & Kratzer, J. (2003). Virtuality, communication, and new product team creativity: a social network perspective. Journal of Engineering and Technology Management, 20(1-2):69-92. Special Issue on Research Issues in Knowledge Management and Virtual Collaboration in New Product Development.

León de la Barra, C. & Crawford, B. (2007). Fostering creativity thinking in agile software development. Lecture Notes in Computer Science, 4799:415-426.

Leonard, D. & Swap, W. (1999). When Sparks Fly: Igniting Creativity in Groups. Harvard Business School Press, Boston.

Lumsdaine, E. & Lumsdaine, M. (1995). Creative Problem Solving: Thinking Skills for a Changing World. McGraw-Hill, New York.

Maiden, N. & Gizikis, A. (2001). Where do requirements come from? IEEE Software, 18:10-12.

Maiden, N., Gizikis, A., & Robertson, S. (2004). Provoking creativity: Imagine what your requirements could be like. IEEE Software, 21:68-75.

Mentzas, G. (2000). The two faces of knowledge man-agement. International Consultant's Guide, pages 10-11. Available at http//imu.iccs.ntua.gr/Papers/O37-icg.pdf.

Mich, L., Anesi, C., & Berry, D. (2005). Applying a pragmatics based creativity fostering technique to requirements elicitation. Requir. Eng., 10:262-275.

Moe, N., Dingsoyr, T., & Dyba, T. (2010). A teamwork model for understanding an agile team: A case

study of a scrum project. Information and Software Technology, 52:480-491.

Neves, F., Correia, A., Rosa, V., & de Castro Neto, M. (2011). Knowledge creation and sharing in software development teams using agile methodologies: Key insights affecting their adoption. In Information Systems and Technologies (CISTI), 6th Iberian Conference, pages 1-6.

Nguyen, L. & Cybulski, J. L. (2008). Into the future: inspiring and stimulating users' creativity. In PACIS, page 203. AISeL.

Nguyen, L. & Shanks, G. G. (2009). A framework for understanding creativity in requirements engineering. Information & Software Technology, 51(3):655-662.

Nonaka, I. & Takeuchi, H. (1995). The Knowledge Creating Company. Oxford University Press, USA.

O'hEocha, C. & Conboy, K. (2010). The role of the user story agile practice in innovation. In Abrahamsson, P. and Oza, N. V., editors, LESS, volume 65 of Lecture Notes in Business Information Processing, pages 20-30.Springer.

Patel, A., Sey, A., Taghavi, M., Wills, C., Na, L., Latih, R., & Misra, S. (2012). A comparative study of agile, component-based, aspect-oriented and mashup software development methods. Technical Gazette, 19(1):175-189.

Robertson, J. (2002). Eureka! why analysts should invent requirements. IEEE Software, 19:20-22.

Robertson, J. (2005). Requirements analysts must also be inventors. IEEE Software, 22:48-50.

Rus, I. & Lindvall, M. (2002). Knowledge management in software engineering. IEEE Software, 19(3):26-38.

Sandmeier, P. (2009). Customer integration strategies for innovation projects: anticipation and brokering. IJTM, 48(1):1-23.

Sandmeier, P. & Gassmann, O. (2006). Extreme innovation; nearly two-thirds of new products fail after launch. The extreme programming methods used in software engineering show how firms can adopt more effective, customer-led innovation processes. European Business Forum, Autumn 2006(26).

Santos, V. A. & Goldman, A. (2011). An approach on applying organizational learning in agile software organizations. In HICSS, pages 4852-4861.

Sanz, L. F. & Misra, S. (2011). Influence of human factors in software quality and productivity. In Murgante, B., Gervasi, O., Iglesias, A.,Taniar, D., and Apduhan, B. O., editors,ICCSA (5), volume 6786 of Lecture Notes in Computer Science, pages 257-269. Springer.

Schwaber, K. & Beedle, M. (2001). Agile Software Development with Scrum. Prentice Hall PTR, Upper Saddle River, NJ, USA, 1st edition.

Sung, S. Y. & Choi, J. N. (2012). Effects of team knowledge management on the creativity and financial performance of organizational teams. Organizational Behavior and

Human Decision Processes, 118(1):4-13.

Takeuchi, H. & Nonaka, I. (1986). The new product development game. Harvard Business Review.

Veryzer, R. W. (1998). Discontinuous innovation and the new product development process. Journal of Product Innovation Management, 15(4):304-321.

von Hippel, E., Thomke, S., & Sonnack, M. (2001). Creating Breakthroughs at 3M. HBR on Point.

Wallas, G. (1926). The art of thought. Harcourt Brace, New York.