Knowledge Management and Creativity in Software Engineering
The Foundations of Agility

Broderick Crawford[1,2], Claudio León de la Barra[1], Ricardo Soto[1,3], Sanjay Misra[4] and 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]*Federal University of Technology, Minna, Nigeria*
[5]*LINA, Université de Nantes, Nantes, France*

Keywords: Knowledge Management, Creativity, Software Engineering, Agile Methodologies.

Abstract: Software development is a knowledge intensive activity and its success depends on knowledge and creativity of the developers. In the last years the traditional perspective on software development is changing and agile methods have received considerable attention. Among other attributes, the agilists claim that fostering knowledge sharing and creativity is one of the keys to response to common problems and challenges of software development today. The development of new software products requires the generation of novel and useful ideas. The purpose of this paper is to provide an understanding of knowledge management and creativity in relation with new software engineering trends. The implications of these findings are considered, and some possible directions for future research are suggested.

1 INTRODUCTION

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. In software organizations the knowledge held by the employees is the main asset and software development projects depends 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 and 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 have emphasized the importance of collaboration and interaction in the software development and, by other hand, creative work commonly involves collaboration in some form and it can be understood as an interaction between an individual and a sociocultural context (Sanz and Misra, 2011). We believe that the innovation and development of new products is an interdisciplinary issue (Takeuchi and Nonaka, 1986)(Nonaka and Takeuchi, 1995), we are interested in the study of the potential of new concepts and techniques to foster knowledge management and creativity in software engineering (Gu and Tong, 2004) (Crawford et al., 2012).

This paper is organised as follows: Section 2 is dedicated to the presentation of Knowledge Management. Section 3 presents the background and general concepts on Creativity. Finally, in Section 4 we conclude the paper and give some perspectives for future research.

2 KNOWLEDGE MANAGEMENT

One of the most widely accepted approaches to classifying knowledge from a KM perspective is the *Knowl-*

DOI: 10.5220/0004447802650272
In *Proceedings of the 15th International Conference on Enterprise Information Systems* (ICEIS-2013), pages 265-272
ISBN: 978-989-8565-60-0
Copyright © 2013 SCITEPRESS (Science and Technology Publications, Lda.)

edge Matrix of Nonaka and Takeuchi (Nonaka and 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: socialisation (from tacit to tacit, whereby an individual acquires tacit knowledge directly from others through shared experience, observation, imitation and so on); externalisation (from tacit to explicit, through articulation of tacit knowledge into explicit concepts); combination (from explicit to explicit, through a systematisation of concepts drawing on different bodies of explicit knowledge); and internalisation (from explicit to tacit, through a process of learning by doing and through a verbalisation and documentation of experiences). Nonaka and Takeuchi model the process of organisational knowledge creation as a spiral in which knowledge is amplified through these four modes of knowledge conversion (See Figure 1).

2.1 Knowledge Management in Software Engineering

The main argument to Knowledge Management in software engineering is that it is a human and 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. Some of the sources of knowledge (artifacts, objects, components, patterns, templates and containers) are stored in electronic form. However, the majority of knowledge is tacit, residing in the brains of the employees. A way to address this problem can be to develop a knowledge sharing culture, as well as technology support for knowledge management. There are several reasons to believe that knowledge management for software engineering would be easier to implement than in other organizations: technology is not be intimidating to software engineers and they believe the tools will help them do a better job; all artifacts are already in electronic form and can easily be distributed and shared; and the fact that knowledge sharing between software engineers already does occur to a large degree in many successful software collaborative projects (Rus and Lindvall, 2002)(Mentzas, 2000)(Apostolou and Mentzas, 2003).

3 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 and Choi, 2012). Althoug 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 and Shalley, 2004)(Chen, 2006).

3.1 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 traditional approaches (often referred to as plan-driven, task-based or Tayloristic), like the waterfall model and its variances, facilitate knowledge sharing primarily through documentation. They also promote usage of role based teams and detailed plans of the entire software development life-cycle. It shifts the focus from individuals and their creative abilities to the processes themselves. In contrary, agile methods emphasise and value individuals and interactions over processes. Tayloristic methods heavily and rigorously use documentation for capturing knowledge gained in the activities of a software project lifecycle (Chau and Maurer, 2004). In contrast, agile methods suggest that most of the written documentation can be replaced by enhanced informal communications among team members internally and between the team and the customers with a stronger emphasis on tacit knowledge rather than explicit knowledge (Beck et al., 2001).

Since human creativity is thought as the source to resolve complex problem or create innovative products, one possibility to improve the software development process is to design a process which can stimulate the creativity of the developers. There are few studies reported on the importance of creativity in

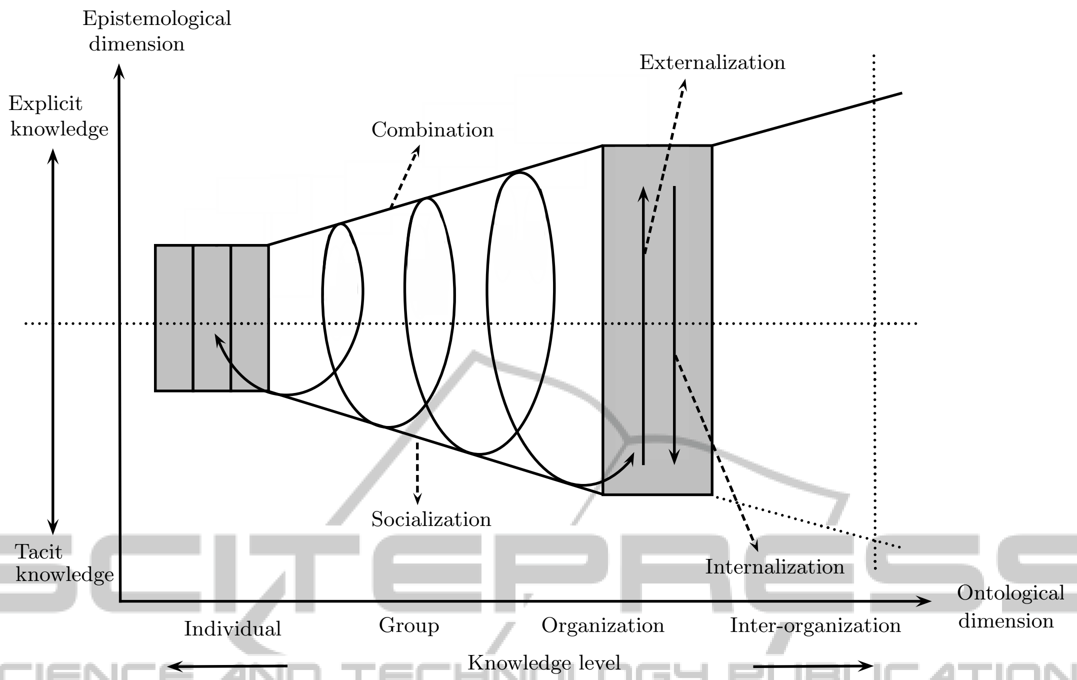

Figure 1: Spiral of Organization Knowledge Creation.

software development. In management and business, researchers have done much work about creativity and obtained evidence that the employees who had appropriate creativity characteristics, worked on complex, challenging jobs, and were supervised in a supportive, noncontrolling fashion, produced more creative work. Then, according to the previous ideas 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 and Tong, 2004)(Glass, 1995)(Crawford and León de la Barra, 2007)(León de la Barra and Crawford, 2007)(Crawford et al., 2008)(Crawford and León de la Barra, 2008) and mostly focused in the requirements engineering (Robertson, 2005)(Mich et al., 2005). 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, 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).

However, in requirements engineering the answers do not arrive by themselves, it is necessary to ask, observe, discover, and increasingly create requirements. If the goal is to build competitive and imaginative products, we must make creativity part of the requirements process. Indeed, the importance of creative thinking is expected to increase over the next decade (Maiden and 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? 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. 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. 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 trans-formational 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. Requirements engineering practitioners have explicitly focused on combinatorial and transformational creativity.

Agile software development is a group of software development methods based on iterative and incremental development, where requirements and solutions evolve through collaboration between self-organizing and cross-functional teams. It promotes adaptive planning, evolutionary development and delivery, a time-boxed iterative approach, and encourages rapid and flexible response to change. It is a conceptual framework introduced in the Agile Manifesto in 2001 (Beck et al., 2001).

3.2 Creative Process

The creative process constitutes the central aspect of team performance, because it supposes a serie of clearly distinguishable phases that had to be realized by one or more of the team members in order to obtain a concrete creative result.

The phases - on the basis of Wallas (Wallas, 1926) and Leonard and Swap (Leonard and Swap, 1999) - are the following ones:

- *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.

- *Encounter:* the findings corresponding to the perception of a problematic situation. For this situation a solution does not exist. It is a new problem.

- *Final preparation:* it corresponds to the understanding and foundation of the problem. It's the immersion in the problem and the use of knowledge and analytical abilities. It includes search for data and the detailed analysis of factors and variables.

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

- *Incubation:* it corresponds to the required time to reflect about the elaborated alternatives, and ”to test them mentally”.

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

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

Considering the creativity as a ”nonlinear” process some adjustments are necessary, redefinitions or discardings that force to return to previous phases, in a complex creative dynamic. Therefore, 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 and Armstrong, 2003)(Leonard and Swap, 1999). By example, in eXtreme Programming (XP) (Beck, 2000) it's important to correlate its phases with the phases considered in a creative process.

- The initial preparation and ”finding” defined in the creative process correspond 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.

3.3 Roles in a Creative Team

Lumsdaine and Lumsdaine (Lumsdaine and Lumsdaine, 1995) raise the subject of the required cognitives abilities (mindsets) for creative problem resolution. Their tipology is excellent for the creative team, and the different roles to consider. These roles are the following ones:

- The *Detective* is in charge of collecting the greatest quantity of information related to the problem. It has to collect data without making judgements, even when it thinks that it has already understood the problem exactly.

- The *Explorer* detects what can happen in the area of the problem and its context. It thinks on its long term effects and it anticipates certain developments that can affect the context (in this case, the team). The explorer perceives the problem in a broad sense.

- The *Artist* creates new things, transforming the information. It must be able to break his own schemes to generate eccentric ideas, with imagination and feeling.

- The *Engineer* is the one in charge of evaluating new ide-as. It must make converge the ideas, in order to clarify the concepts and to obtain practical ideas that can be implemented for the resolution of problems.

- The *Judge* must do a hierarchy of ideas and decide which of them will be implemented (and as well, which ones must be discarded). Additionally, it must detect possible faults or inconsistences, as well as raise the corresponding solutions. Its role must be critical and impartial, having to look for the best idea, evaluating the associated risks.

- The *Producer* is in charge of implementing the chosen ideas.

Leonard and Swap (Leonard and Swap, 1999) have mentioned additional roles, possible to be integrated with the previous ones, because they try to make more fruitful the divergence and the convergence in the creative process:

The *provoker* who 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/her experticia and experience.

The *facilitator* whose function consists in helping and supporting the team work in its creative task in different stages.

The *manager* who cares for the performance and specially 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 raised a role tipology similar to Lumsdaine and Lumsdaine (Lumsdaine and Lumsdaine, 1995), being interesting that they group the roles in three categories: those directed to the learning of the creative team (susceptible of corresponding with the detective, explorer, artist, provoker and think tank roles), others directed to the internal organization and success of the team (similar to the judge, facilitator and manager roles) and, finally, roles whose purpose is to construct the innovation (possibly related to the role of the engineer and judge).

By example, the following is the correlation between creative and XP roles:

- The detective function consisting in collecting information related to a problem is made by the client him-self in XP, because this one 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 they appreciate the reach of the identified problem, as well as of the possible solutions.

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

- 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.

- The function of the judge, understood as the definitive selection of the solutions to implant, is made in XP by the tracker and the client.

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

The supporting roles considered are:

- The provoker; creativity demands that the divergence as well as convergence in the solutions be maximum and complete. There is not explicit reference in XP methodology about divergent thinking.

- The think tank who helps the team work ”from outside” is equivalent completely to the role of the consultant.

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

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

3.4 Basic Organizational Conditions

Regarding to the structure dimension of a new product development team (in particular software), it is possible to relate the roles in creativity to the roles defined in the agile methodology distinguishing: base roles, that is, 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. In relation with the structure dimension it’s 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's inserted (Amabile, 1998)(Isaksen et al., 1999)(Leonard and Swap, 1999). Some conditions are necessary - although not sufficient - for the creative performance. We are interested in explore the influence of autonomy, communication, cooperation and learning, the handling of possible conflicts, pressure, formalization, performance evaluation, available resources (time) and the physical atmosphere of work.

The *autonomy* refers to the capacity of the people and the team as a whole to act and make decisions. By example, this aspect is related to the following XP practices: the actual client, 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 team member's *communication*, cooperation and learning are fortified since the client is present and there exist opened spaces to work together and in a pair programming mode. The work dynamics is based on a game of planning 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, 40 hour a week and codification standard). Cooperation and trust are associated to this issue. The *pressure* (that in creativity is appraised as favorable until certain degree, favoring the performance, and detrimental if it exceeds this degree), is susceptible then to favor in XP through the client in situ, the programming by pairs, the planning game, the tests and continuous integration. It's possible to avoid, or at least to reduce, the pressure through the refactorization, the small deliveries, the collective property, and the fact that surpassing the 40 weekly working hours is seen like an error. The *formalization*, that gives account of all those formal aspects (norms, procedures) defined explicitly and that are known, and even shared, by the members of the team. It's assured in XP through planning game, metaphors, continuous integration, the collective property, the 40 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 40 weekly hours (as a nonexceedable metric indicating limit of effective-ness), all at the light of the planning (including the standards). Finally, the presence of client constitutes the permanent and fundamental performance evaluation of the team and the products. The evaluation characteristics empower the learning processs. 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 to optimize the partial working-times, as well as the whole project time, ensuring a positive pressure. The *physical atmosphere of work*, referred in creativity to the surroundings that favor or make difficult the creative performance (including aspects like available spaces, noise, colours, 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.

4 CONCLUSIONS

In Software Engineering many development approaches work repeating the basic linear model in every iteration, 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 emphasis 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.

The use of software tools for rapid prototyping of algorithms would save considerable resources. It would be desirable to employ Agile principles and reusability to produce software which is easily adaptable to changing requirements while also improving the quality and reduce development efforts. Since software development is a knowledge intensive activity, an understanding from a Knowledge Management perspective offers important insights about reusability and Software Engineering methods.

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 wich 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.

The Extreme Programming methodology includes implicitly central aspects of a creative teamwork. These aspects can be organized according to the structure that the team adopts and the performance that characterizes to the team. The structure that the team adopts and specially the different roles that the methodology advises to define, nearly correspond with the roles at the interior of a creative team. The performance that characterizes the team through certain advisable practices, from the perspective of creativity, constitutes the necessary basic conditions, although nonsufficient, in order to favor the group creative performance. These conditions, called practices in XP methodology, are accompanied by concrete phases of constituent activities of an agile software development process, which is possible to correspond with the creative process, which is fundamental to the creative performance. In spite of the previous comments, we think that XP methodology should have a more explicit reference to the provoker role that is thoroughly described in creativity as a fundamental factor to generate innovation. This can be explained because, in general, agile methodologies do not aim, as a central element, to generate an original software, but an effective one. Secondly, it is essential the distinction and formalization of the creative phases to generate options incubation and option choices. It is assumed that they take place in the iterative and production process, although XP is not focused in ”originality”. Thirdly, a more direct mention to the physical atmosphere of work, that in creativity are considered as highly relevant to enhance the performance. These aspects should have a greater consideration since software development is a special case of product development.

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. and 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., Beedle, M., Bennekum, A. V., Cockburn, A., Cunningham, W., Fowler, M., Grenning, J., Highsmith, J., Hunt, A., Jeffries, R., Kern, J., Marick, B., Martin, R. C., Mellor, S., Schwaber, K., Sutherland, J., and 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. and 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 Artificial Intelligence*, pages 173–183. Springer.

Chau, T., Maurer, F., and 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.

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

Crawford, B. and León de la Barra, C. (2008). Integrating creativity into extreme programming process. In Cordeiro, J. and Filipe, J., editors, *ICEIS (3-1)*, pages 216–219.

Crawford, B., León de la Barra, C., and Rubio, J. (2008). 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., and Monfroy, E. (2012). Knowledge management and creativity practices in software engineering. In Liu, K. and Filipe, J., editors, *KMIS*, pages 277–280. SciTePress.

Gilson, L. L. and 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. and Tong, X. (2004). Towards hypotheses on creativity in software development. *PROFES*, 3009:47–61.

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

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

Kelley, T. and 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. and Armstrong, G. (2003). *Principles of Marketing*. Prentice Hall, New Jersey.

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

Leenders, R. T., van Engelen, J. M., and 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. and Crawford, B. (2007). Fostering creativity thinking in agile software development. *Lecture Notes in Computer Science*, 4799:415–426.

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

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

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

Maiden, N., Gizikis, A., and 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 management. *International Consultant's Guide*, pages 10–11. Available at http//imu.iccs.ntua.gr/Papers/O37-icg.pdf.

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

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

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. and Lindvall, M. (2002). Knowledge management in software engineering. *IEEE Software*, 19(3):26–38. Available at http://fc-md.umd.edu/mikli/RusLindvallKMSE.pdf.

Sanz, L. F. and 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.

Sung, S. Y. and 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. and Nonaka, I. (1986). The new new product development game. *Harvard Business Review*.

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