The Influence of New Product Development on Scrum Practices

Lane, Michael T., LERO1 - The Irish Centre for Software Engineering, University of Limerick, Limerick, Ireland, michael.lane@ul.ie

Fitzgerald, Brian, LERO1 - The Irish Centre for Software Engineering, University of Limerick, Limerick, Ireland, brian.fitzgerald@ul.ie

Ågerfalk, Pär J., Department of Information Science, Uppsala University, Uppsala, Sweden, par.agerfalk@im.uu.se

Abstract

One of the most widely used Agile software development (ASD) methods is Scrum. The underlying inspiration for Scrum came from an examination of new product development (NPD). This work proposes that the combination of six management characteristics results in very positive and effective product development: built-in instability, self-organizing project teams, overlapping development phases, "multi-learning", subtle control and organizational transfer of learning (Takeuchi et al. 1986). This paper reports on one strand of an overall research study investigating the motivations for adoption of Scrum. The strand in question focuses on the identification of the particular Scrum practices that relate to NPD characteristics.

1. Introduction

Takeuchi et al. (1986) leverage the metaphor of a rugby team operating as a unit to move a ball forward to characterize effective new product development(NPD). Their observations formed the basis for the information systems development (ISD) method "Scrum". In the modern professional game of rugby it is not unusual to see dominant forward rugby packs forego the opportunity to kick a penalty (worth three points), electing instead to form a scrum to push the ball (and the other team) over the end line to score a "try" and be awarded five points. Scrum is an agile software development (ASD) project management method that is popular in industry (Dyba et al. 2008). However, when an ISD organization elects to apply scrum, do they consider what goal they are trying to achieve? It may not always be appropriate to elect for a scrum in rugby (especially if your goal is to score points and the combined weight and pushing power of your players is inferior to the opposition). In this case, you may deem it preferable to kick a penalty or attempt to run and pass the ball. This paper examines Scrum to determine what goals are achieved by the performance of particular Scrum practices.

This paper opens with a description of the fundamental NPD principles that underpin the Scrum method (table 1). Scrum is introduced by presenting a set of core practices gleaned from a review of authoritative publications on this framework (table 2). These papers range from the first scrum

publication(Schwaber 1995), through a much-cited review of agile methods (Highsmith 2002) up to a recent scrum guide published by the initial founders of scrum(Sutherland et al. 2011). This is followed by a description of a number of supplementary scrum practices that may be applied to perform the core scrum practice set. Each supplementary scrum practice is analysed and associated with appropriate NPD principles.

2. Scrum

The use of the tactics of rugby union as an analogy to describe an innovative form of new product development (Takeuchi et al. 1986) inspired Jeff Sutherland and Ken Schwaber to create the Scrum development method.

The underlying inspiration for Scrum came from an examination of new product development (NPD). This work proposes that the combination of six management characteristics results in very positive and effective product development: built-in instability, self-organizing project teams, overlapping development phases, "multi-learning", subtle control and organizational transfer of learning(Takeuchi et al. 1986). Setting challenging targets and releasing control of how these targets should be met introduces a tension and instability that empowers teams, promoting creativity and innovation. Self-organization incorporates autonomy, self-transcendence and cross-fertilization. Autonomy empowers the team to manage their situation without external interference. Self-transcendence is evident when teams practice continuous improvement via the setting and extending of goals in pursuit of the higher-order challenges set down by senior management. Cross-fertilization occurs as teams share knowledge. (Earlier agile settings promoted co-located teams. It is likely that a motivation for such a configuration was the dissemination of tacit knowledge via socialization(Turban et al. 2007)). Cross-fertilization is both leveraged and augmented by overlapping development phases that promote collective responsibility and the reduction of potential delays. Such delays may be due to bottlenecks and misunderstandings associated with the "over the wall" hand-off approach associated with division of labor using separate specialized development phases.

ID	Principle
NPD-1	Built-in instability
NPD-2	Self-organizing project teams
NPD-3	Overlapping development phases
NPD-4	Multi-learning
NPD-5	Subtle control
NPD-6	Organizational transfer of learning

Table 1: New Product Development principles that inspired Scrum founders (Takeuchi and Nonaka, 1986, Schwaber, 1995)

Multi-learning encompasses the notion of learning being promoted at multiple levels (individual, group, and corporation) and that individuals seek learning across functions (such as ISD and

marketing). Subtle control refers to the management of self-organizing teams by permitting freedom within boundaries that are marshaled through non-invasive management techniques that are tolerant and reflective of the uncertain non-linear nature of new product development. Organizational dissemination of learning is enabled through standardization and sharing via formal artifacts (e.g. project post-mortem (PMI 2008)) and judicious assignment of experienced individuals to appropriate projects. (Takeuchi et al. 1986)

The Scrum method was devised to address concerns about the unpredictable nature of ISD. It is a framework of simple rules that enable continuous inspection and adaptation. The lifecycle of a Scrum project is comprised of a series of development "sprints" or iterations that are bounded by an initial planning phase and a final closure phase (figure 1). Planning enables both architectural and scope concerns to be addressed and the closure phase incorporates release management. The initial and final phases are predictable and theoretical. The empirical nature of Scrum is evident in the ongoing iterations or "sprints" that adapt to feedback and change throughout the project. This nature relies upon transparency, inspection and adaptation. Transparency stipulates that the process is understood and that a standard language is used and accepted by team members and observers. Inspections of artifacts are performed to uncover defects or invalid variances and adaptations to the process are performed if deemed necessary. In core scrum practices, inspection and adaptations may occur at the Sprint planning meeting, daily scrum, sprint review meeting and the sprint retrospective. The method embraces change by enabling the development team to mange change as the information system evolves. (Sutherland et al. 2007a; Sutherland et al. 2011)

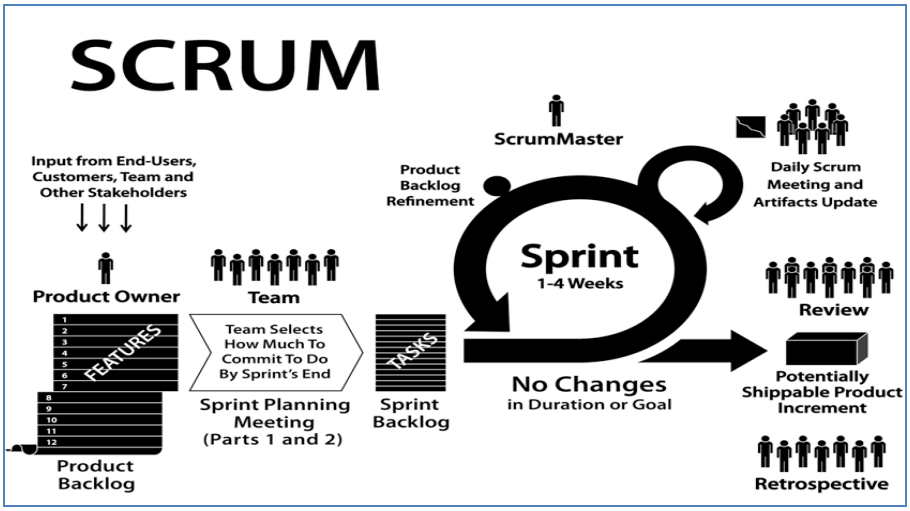

Figure 1: Scrum Framework (Deemer et al., 2010)

In his introductory publication of the Scrum method, (Schwaber 1995) invokes complex adaptive systems theory (CAS) stating that Scrum enables a team to maintain order within the development process while embracing change within the environment. Wang et al. (2009) specify this relationship further by comparing particular CAS concepts with aspects of agility. They state that the notion of inter-connected autonomous agents that can "flock together" to react and resolve an external force is analogous to an ASD team that is empowered to direct its activities and consists of individual members that share their experiences and skills. The team is

then in a position to come together in order to resolve unforeseen issues (i.e. adapt to change). This is consistent with the aforementioned characteristics of self-organization and overlapping development phases. A self-directed ASD team consisting of autonomous individual members who share each other's experiences and skills reflects autonomy and cross-fertilization. Grouping together to resolve development issues indicates an ability to overlap development phases as analysis/design/code and test may be applied to overcome an issue. The CAS concept of "edge of chaos" is described as that state where a team (or organization) can experiment and adapt to situations freely but retain a limited structure in order to avoid falling into disarray. This equates to the aforementioned characteristics "built-in instability" and "subtle control". Finally, "emergence" relates to continuous improvement or new learning leading to new behavior that indicates that a team has adapted and evolved. This concept relates to "multi-learning" and when extended beyond the team may also apply to "organizational transfer of learning".

Sutherland et al. (2011) recommend Scrum as an effective container for other methods. Fitzgerald et al. (2006) describe the effective use of Scrum to plan and manage appropriate XP technical practices. Although the findings of this study question the premise that the implementation of agile methods should not be decomposed into piecemeal selection of practices, it reports the effective tailoring of Scrum and XP differently. Scrum tailoring involves the use of additional or supplementary practices to enrich the core set of practices. XP tailoring results in the application of a subset of the core set of practices. The founders of Scrum state that although it is possible to pick and choose certain core practices or omit certain roles, the Scrum framework should be used in its entirety. In cases where roles or practices have been omitted, the resultant framework should not be called Scrum(Sutherland et al. 2011).

The position taken in this paper is that the use of supplementary practices to deepen consideration of each core Scrum role and practice upholds the guideline of using the complete framework. A set of "core" scrum practices are presented in table reflecting an analysis of published reviews of Scrum from its first introduction(Schwaber 1995), through a major ASD contribution(Highsmith 2002) and up to a recent definition of the framework by its founders (Sutherland et al. 2011). Each core scrum practice is identified by an ID prefixed by "CS". CS-7 was initially identified as "Daily Scrum" but has been broadened by the authors to reflect general Sprint management activities including Daily scrum. An additional core practice, CS-8 "Sprint-technical activities" was not proposed by any of the sources (technical practices are not explicitly called out by Scrum). It was added to reflect the fact that in order to do ISD using Scrum, technical activities must be performed. Certain technical practices support Scrum goals and it is deemed useful to include this category as a core scrum practice in order to more fully explore these goals.

The core scrum practices enable a project to exhibit the characteristics of innovative new product development. An established team configuration promotes subtle control and team self-organization. A product owner provides the "single point of accountability" for product features, shielding the team from interference by multiple stakeholders. The scrum-master promotes team self-organization by guiding without directing.

ID	Practice	Description
CS-1	Scrum Team Configuration	A scrum team comprises a product owner, scrum-master and developers. This team are cross-functional and self-organizing
CS-2	Initial planning (and ongoing) - create or update product backlog	Create product backlog - list of features needed in product.
CS-3	Pre-Sprint planning - create release backlog	Determine list of features to be included in first release of product
CS-4	Pre-Sprint planning - create sprint backlog	Determine features to be developed in Sprint. Determine and estimate tasks to be performed.
CS-5	Pre-Sprint planning - define sprint goal	High-level business purpose that provides rationale for Sprint features - much like a project charter (PMI, 2008)
CS-6	Sprint - lock features for duration	Disallow any external modifications during Sprint. Introduce "stability zone". Promote careful planning.
CS-7	Sprint - daily scrum meeting and other general activities	Short meeting to identify and plan the removal of any impediments to team's work. Enable information sharing and learning. Monitor progress.
CS-8	Sprint - technical activities	Although technical practices are not called out as part of the framework, certain technical practices support many Scrum goals
CS-9	Sprint - sprint backlog graph (burndown chart)	X-axis = Days of sprint; Y-axis = work-hours remaining; Simple visual control of work remaining. Updated every day following re-calibration of remaining time after daily scrum. Tracks downward to zero
CS-10	Post-Sprint - review meeting	Demonstrate features constructed. Review technical work completed
CS-11	Post-Sprint - team retrospective	Inspect how team performed in Sprint. Similar to project "lessons learned".
CS-12	Definition of Done	All the team agree on what constitutes a "done" product increment. Item must be "done" to be potentially shippable. Team may increase done qualification criteria as it matures.

Table 2: Proposed Core Scrum practices from literature review and analysis (Schwaber 1995, Highsmith 2002, Sutherland andSchwaber 2011)

The development team contains a heterogeneous set of skills. However the only role permitted in this team is "developer" thus promoting the characteristic of overlapping phases. Practices such

as the creation of sprint backlog and daily scrum promote and facilitate self-organization among the development team. Permitting the team to set the amount of work they may achieve in a sprint, combined with feedback mechanisms such as sprint backlog graph and sprint review facilitate a team's understanding of their velocity and have the potential to encourage self-transcendence as they accelerate their velocity. Freezing requirements within a sprint exhibits subtle control while permitting unlimited change to the product backlog builds in controlled instability by permitting the team to work in an orderly way within an extremely volatile environment. Sprint reviews promote transfer of information to other parts of the organization as this meeting informs stakeholders about the increment from a technical and business perspective. Team retrospective promotes multi-level learning as individuals reflect upon their performance and the team's performance. Arguably, the Sprint retrospective may lead to process improvements and updated standards, thus enabling transfer of information to other parts of the organization. Such behavior is not called out currently in retrospective activities although a variant practice associated with retrospectives (Sprint retrospective with cross-pollination) does promote information dissemination to the wider organization.

3. Supplementary Scrum practices.

In order to probe further into the Scrum method, additional detail on the application of the above core practices is supplied by examining contributions from various sources. In many cases, these supplementary practices represent a variation on one of the core practices outlined above. It should be noted that any list of supplementary practices is unlikely to be exhaustive. A synthesis of additional Scrum practices is derived from an iterative analysis process of practice sets gleaned from papers and books that examine the Scrum framework. The analysis approach addressed general considerations associated with Scrum and practices associated with different phases of a Scrum project lifecycle.

3.1 General Considerations

A key tenet of Scrum is its transparency and this is supported by simple rules to promote consistent comprehension and application of the framework. Cohn (2010) highlights the need for small cross-functional teams, citing studies claiming evidence of "hiding" or "social loafing" when individuals feel that somebody else can cover for them. Cohesion is promoted in smaller teams due to increased interaction. The allocation of all development activities related to a particular feature set to a team can prove effective. Such encapsulation of feature development offers potential to achieve the same advantages as use of this concept can provide in object-oriented development: "high-cohesion" and "low-coupling". Clarity of purpose can lead to increased commitment to value creation. Reduced dependencies on component teams and their associated sprints reduce the risk of delays due to integration difficulties or planning challenges and bottlenecks. In order to nurture self-organization, selection and de-selection of team members is critical. Cross-fertilization demands consideration of diverse technical and domain skills. This is one aspect of the "subtle control" characteristic that may be employed by management to promote effective self-organization. Within the scrum environment, subtle control may be practiced by the Scrum-master in how he/she engages with the dynamics within the development team. Cohn (2010) presents a practice that leverages a model of interconnected concepts in self-organization proposed by (Eoyang 2001): containers, differences and transforming exchanges (CDE). This stipulates that if a scrum-master is aware of issues within a

team, he/she may alter the "container" bounding a development team in order to influence how it self-organizes. Alteration of containers can range from a physical change such as the reduction of cubicle walls to a less tangible change such as the inclusion of the team in a community of practice.

Differences within a team influence the manner in which it self-organizes so increasing or decreasing differences can impact this process e.g. asking probing questions in order to promote constructive dissent. Finally transforming exchanges refer to the manner in which a team self-organizes to address a challenge. For example, continuous speeches related to concerns about product marketability by a product manager may result in either motivating or de-motivating a team depending upon the situation. Altering the frequency of such an exchange may have a positive impact on the team's ability to self-organize. Having team's work together for a long time and dedicated to one project promotes relationship building as well as reducing waste caused by juggling multiple conflicting priorities. Learning occurs at the "team" level as increased understanding and commitment to working together within a domain leads to increased performance (Tuckman 1965). From a project management perspective, such a practice resembles a "projectized" team configuration and is recommended for R&D projects (PMI 2008).

3.2 Initial and pre-sprint planning

Initial planning of a Scrum project involves establishing a shared understanding within the team about initial goals, rationale and vision for the project. The product backlog is used to facilitate this process. The dynamic nature of this requirements list is reflected in its continual refinement throughout the project.

3.3 Sprint

Before sprint commencement, the core practice of "sprint planning meeting" is used by the Scrum team to pull items from the top of the product backlog and determine how they will be completed. Sutherland et al. (2011) recommend that this be a two-part meeting to determine the "what" and "how" of the forthcoming sprint. Firstly, the development team select items to be developed from the product backlog. In the second part of the meeting, the development team conducts a high-level design in order to be confident that the work can be achieved within the sprint. This self-organization enables the team to maintain a sustainable workload. Establishment of a "sprint goal" by the development team completes sprint planning. This focuses the team on the rationale for delivering the product increment at the end of the sprint. Comprehension of the overarching goal to be achieved can help a team develop and sustain a mode of effective performance(Tuckman 1965). From a planning and stakeholder management perspective, the sprint goal may be a milestone in a product roadmap.(Sutherland et al. 2011)

The heart of Scrum is a Sprint, a time-box of one month or less during which a "Done", useable, and potentially releasable product Increment is created.

(Sutherland et al. 2011)

The core practices of daily scrum and sprint backlog graph enable continual inspection and adaptation during development. Consistent sprint lengths enable the team to gain a better understanding of its capacity to deliver user stories. This enables it to better plan and meet stakeholder expectations. Although it is good practice to consistently test features as a product is

being built, it is wise to consider carefully how the backlog items will suit this approach. Mixing backlog items to ensure that certain items are completed early in the sprint can help avoid bottlenecks or handoff delays near the end of the sprint. The visual progress chart (or variations thereof) can assist in exposing an increasing number of commenced but incomplete activities. A more advanced form of Scrum that incorporates lean principles and encourages flow extends this concept further. The introduction of a Kanban or pull approach combined with limits of how many items may be in a particular stage (such as "in-progress") can force the team to resolve bottlenecks that carry the risk of many items failing to be completed at the end of the sprint(Ikonen et al. 2011; Kniberg et al. 2010). Acceptance test driven development scopes the development of an item by linking the user story to the conditions of satisfaction specified by the product owner when considering the item as part of a releasable product increment. Although the product owner can change items on the product backlog at any time, it is also useful for the development team to be involved in refining ("grooming") the backlog to add detail such as estimates or user story clarifications.

3.4 Sprint - technical practices

Although Scrum is a framework used to manage other methods, certain technical practices are often associated with Scrum projects. Developing a test prior to designing and coding a user story assists the development team in fully comprehending the work to be performed. Pair programming promotes collective ownership and improves quality by advocating better designs through sharing of abstractions and linking analysis to design(Wasserman 1996), continuous peer review and assistance in identifying and implementing refactoring opportunities. Such an approach facilitates the overlapping of phases and favours face-to-face discussion. It is likely that both peer review and refactoring should result in peer-learning. Product backlog items or associated user-stories should be allocated with clear interfaces as more detail emerges on these items. This low-coupling of "packets" of work can ease design, testing and maintenance (Cohen et al. 2004). Such encapsulation can be seen as subtle control as teams are obligated to conduct design-by-interface but are empowered to evolve their own feature thus following an agile principle that the best architectures and designs emerge from self-organizing teams.

3.5 Post-Sprint Activities

The core practices of sprint review and sprint retrospective enable inspection and adaptation by reflecting upon both the product and team performance. Insights uncovered from this work may then lead to product backlog updates or modified team practices. Using a scrum-master from another team to act as a facilitator for a scrum retrospective may prove useful in disseminating information throughout the organization. A general technique for sprint retrospectives is to populate a two-column chart listing activities that "went well" and "what could be better". A variation on this technique is to annotate items with a "c" if they were caused by Scrum and a "v" if they were made visible by scrum. This has the potential to reinforce Scrum as a useful technique for causing good work performance and exposing performance deficiencies (Deemer et al. 2010). The NPD principle of multi-learning is enabled by this practice. An oft-cited core practice of Scrum is the ability to call the project done after any sprint i.e. every sprint produces a shippable product increment that is fully integrated with all prior increments. However, the ability to be done after any sprint is an ideal approach that may not always be possible due to

weaknesses in an organization's infrastructure and processes. In these cases, a release sprint may be necessary to "wrap up" and coordinate release concerns.

4. Conclusions and recommendations

This paper opens with a description of the fundamental NPD principles that underpin the Scrum method (Table 1). Scrum is introduced by presenting a set of core practices gleaned from a review of authoritative publications on this framework (table 2). Section 2 presents a number of supplementary scrum practices that offer more detail around variant practices that may be applied to perform the core scrum practice set. Each supplementary scrum practice is analysed and associated with appropriate NPD principles.

It is acknowledged that ASD in general and Scrum in particular require further research, especially in the areas of scaling Scrum and the application of this method in GSD (Abrahamsson et al. 2009; Dyba et al. 2008). A clear understanding of the motivations behind the application of particular Scrum practices can facilitate this research. An established set of practice-goal mappings could be used as a lens to conduct investigations into such contexts as scaling Scrum to large teams or the application of Scrum in global software development (GSD). Such empirical investigations will lead to a greater understanding of how this method is applied "in-action" and assist in research into mechanisms on tailoring these methods to best meet different situational needs. For example, particular Scrum practices that have emerged to address GSD situations could be "rolled up" to their associated core Scrum practice in order to investigate the motivations guiding the implementation of particular GSD practices.

5. Acknowledgements

This work was supported, in part, by Science Foundation Ireland grant 10/CE/I1855 to Lero - the Irish Software Engineering Research Centre (www.lero.ie)

6. References

Abrahamsson, P., Conboy, K., and Wang, X. "`Lots done, more to do': the current state of agile systems development research," *European Journal of Information Systems* (18:4) 2009, pp 281-284.

Abrahamsson, P., Warsta, J., Siponen, M. T., and Ronkainen, J. "New directions on agile methods: a comparative analysis," in: *Proceedings of the 25th International Conference on Software Engineering*, IEEE Computer Society, Portland, Oregon, 2003, pp. 244-254.

Agerfalk, P. J., Fitzgerald, B., Holmstrom, H., Lings, B., Lundell, B., and O. Conchuir, E. "A Framework for Considering Opportunities and Threats in Distributed Software Development " in: *International Workshop on Distributed Software Development,*, Austrian Computer Society, Paris, 2005, pp. 47--61.

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

Carmel, E. *Global Software Teams: Collaborating Across Borders and Time Zones* Prentice Hall., Upper Saddle River, 1999.

Cohen, D., Lindvall, M., and Costa, P. "An introduction to agile methods.," *Advances in Computers, Advances in Software Engineering* (62:66) 2004, pp 2-67.
Cohn, M. *SUCCEEDING WITH AGILE Software Development Using Scrum*, (1 ed.) Pearson Education Inc., Boston MA, 2010.
Conboy, K., and Fitzgerald, B. "Toward a Conceptual Framework of Agile Methods," in: *Extreme Programming and Agile Methods - XP/ Agile Universe*, Springer-Verlag, Berlin, 2004.
Cristal, M., Wildt, D., and Prikladnicki, R. "Usage of Scrum practices within a global company.," ICGSE 2008, Bangalore, India, 2008, pp. 222-226.
Daft, R. L., and Lengel, R. H. "Information richness: a new approach to managerial behavior and organizational design," *Research in organizational behavior* (6) 1984, pp 191-233.
De Souza, C. R. B. "Global Software Development: Challenges and Perspectives ", 2001.
Deemer, P., Benefield, G., Larman, C., and Vodde, B. "The Scrum Primer," 2010.
Dodda, S., and Ansari, R. "The Use of SCRUM in Global Software Development: An Exploratory Study," in: *School of Computing*, Blekinge Institute of Technology, SE - 371 39 Karlskrona, 2010.
Dyba, T., and Dingsoyr, T. "Empirical studies of agile software development: A systematic review," *Inf. Softw. Technol.* (50:9-10) 2008, pp 833-859.
Eoyang, G. H. "Conditions for Self-organizing in Human Systems," The Union Institute and University, 2001.
Fitzgerald, B., Hartnett, G., and Conboy, K. "Customising agile methods to software practices at Intel Shannon," *European Journal of Information Systems* (15:2), Apr 2006, pp 200-213.
Fowler, M., and Highsmith, J. "The Agile Manifesto," *Software Development* (9:8) 2001, pp 28-32.
Gannon, S. "A Case Study on Software Release Management for Scrum Projects within a System Release.," in: *J.E. Cairnes School of Business and Economics*, National University of Ireland, Galway, Galway, Ireland, 2011.
Grinter, R. E., Herbsleb, J. D., and Perry, D. E. "The geography of coordination: dealing with distance in R&D work," in: *Proceedings of the international ACM SIGGROUP conference on Supporting group work*, ACM, Phoenix, Arizona, United States, 1999, pp. 306-315.
Highsmith, J. *Agile Software Development Ecosystems* Addison-Wesley, 2002.
Hossain, E., Babar, M. A., and Hye-young, P. "Using Scrum in Global Software Development: A Systematic Literature Review," Fourth IEEE International Conference on Global Software Engineering, 2009, pp. 175-184.
Ikonen, M., Pirinen, E., Fagerholm, F., Kettunen, P., and Abrahamsson, P. "On the Impact of Kanban on Software Project Work An Empirical Case Study Investigation," 16th IEEE International Conference on Engineering of Complex Computer Systems (ICECCS), IEEE, Las Vegas, NV, 2011, pp. 305-314.
Kniberg, H. "Scrum and XP from the Trenches," D. Plesa (ed.), C4Media, USA, 2007.
Kniberg, H., and Skarin, M. "Kanban and Scrum - making the most of both," C4Media Inc., USA, 2010.
O'Conchuir, E., Par, J. A., Olsson, H. H., and Fitzgerald, B. "Global software development: where are the benefits?," *Commun. ACM* (52:8) 2009, pp 127-131.
PMI *A Guide To The Project Management Body Of Knowledge (PMBOK Guides)*, (4th ed.) Project Management Institute, 2008.

Pries-Heje, L., and Pries-Heje, J. "AGILE & DISTRIBUTED PROJECT MANAGEMENT: A CASE STUDY REVEALING WHY SCRUM IS USEFUL" " ECIS 2011, Helsinki, Finland, 2011.

Schwaber, K. "Scrum Development Process," in: *OOPSLA'95 Workshop on Business Object Design and Implementation,* J.Sutherland, D. Patel and J. Miller (eds.), Austin, TX, 1995.

Sutherland, J., and Schwaber, K. "The Scrum Papers: Nuts, Bolts, and Origins of an Agile Method. ," Scrum Inc., Boston, 2007a.

Sutherland, J., and Schwaber, K. "The Scrum Guide," http://www.scrum.org/scrumguides, 2011.

Sutherland, J., Viktorov, A., Blount, J., and Puntikov, N. "Distributed Scrum: agile project management with outsourced development teams," in: *Hawaii International Conference on Software Systems (HICSS'40)*, Big Island, Hawaii, 2007b.

Takeuchi, H., and Nonaka, I. "The New New Product Development Game," *Harvard Business Review*) 1986.

Tuckman, B. "Developmental sequence in small groups," *Psychological Bulletin* (63) 1965, pp 384-399.

Turban, Leidner, McLean, and Wetherbe *Information Technology for Management: Transforming Organizations in the Digital Economy, 6th ed.*, (6th ed.) Wiley, New York, NY, 2007.

Wang, X., and Conboy, K. "Understanding Agility in Software Development through A Complex Adaptive Systems Perspective," 17th European Conference on Information Systems, Verona, 2009.

Wasserman, A. I. "Toward a Discipline of Software Engineering," *IEEE software* (13:6) 1996, pp 23-31.

Woodward, E., Surdek, S., and Ganis, M. *A Practical Guide to Distributed Scrum* Pearson Education Inc., Boston Ma, 2010.