Comparing Apples with Oranges? The Perspectives of a Lean Online Community on the Differences between Agile and Lean

Research-in-Progress

Xiaofeng Wang
Free University of Bozen
Bozen, Italy
xiaofeng.wang@unibz.it

Kieran Conboy
University of New South Wales
Sydney, Australia
k.conboy@unsw.edu.au

Abstract

The evolution processes of agile and lean software development are intertwined. Recently a shift from agile methods to lean software development has been noticed and advocated. To truly comprehend this phenomenon theoretically and practically, we take a step back and investigate the key differences between agile and lean software processes, the understanding of which is still limited. A content analysis of a popular mailing list of the lean community, called kanbandev, is being conducted. The results of the study unveil the perspectives of the lean community on the differences between agile and lean software development processes.

Keywords: Lean software development, lean thinking, agile methods, kanban, kanbandev

Introduction

Lean manufacturing was incepted in the Toyota Production System (TPS) around the 1940s, even though some of the concepts were actually used prior to this by the Ford Automobile company[1]. It revolutionised the automobile manufacturing industry and has been copied and extended around the world (Womack et al. 1990). The core principles underlying lean manufacturing have not been confined to manufacturing processes but have been applied in other disciplines too. When the Agile Alliance was founded and the the Agile Manifesto was published in 2001. Lean thinking was one of the inspiring sources (Highsmith 2002).

In the last several years, even though eXtreme Programming (XP) and Scrum are still dominating agile software development as the most popular methods (VersionOne 2010), lean software development as a distinct software development methodology has been noticed and advocated. Several experience reports published in the agile conferences show the evidence of the transition from agile to lean processes (Willeke 2009, Rutherford et al. 2010, Birkeland 2010). Dedicated lean conferences are emerging (http://www.leanssc.org/). Rooted in lean thinking, lean software development has incorporated various ideas and elements of agile methods.

Due to the intertwined evolution of agile and lean approaches, a natural and intriguing question to ask is: are agile and lean just two different names for the same thing, or they are actually different therefore the challenges and issues faced by modern software companies when using agile processes (and noting the seriousness of these issues) are addressed by lean approaches? The theoretical motivation of our study is to understand whether lean software development is turning into a new "dominant" methodology for development, or merely an iteration of current agile approaches? To be able to answer these questions, we need to take a step back and understand the key differences between agile and lean software processes.

Since the research on the broad topic of lean software development is still a nascent area (Dingsøyr et al. 2008), there is yet a good understanding of lean software processes, even to a less extent the differences between agile and lean processes. The purpose of our study is to gain a better understanding of the key differences between agile processes and lean processes. As a first step towards this end, we conduct an analysis of a popular mailing list of the lean community, called "kanbandev"[2], which allows us to understand the perspectives of the lean community on the differences between agile and lean processes. We were in the middle of data analysis process at the time the paper was written.

This research in progress paper is organized as follows. The next section reviews the relevant literature on agile and lean to show what we have already known about the differences between agile and lean software processes. This will lead to the construction of a theoretical framework that can be used to frame the subsequent data analysis. The research approach section explains the rationale of the mailing list analysis, describes the data reduction and analysis procedures. The paper ends with the expected results and concluding remarks.

Agile and Lean: Apples and Oranges?

Agile Software Development

In agile literature, agile methods generally mean a family of methods under the umbrella of the Agile Alliance, including XP (Beck 1999), Scrum (Schwaber and Beedle 2002), Dynamic Systems Development Method (DSDM, http://www.dsdm.org), Crystal Methods (Cockburn 2001), Feature-Driven Development (FDD, Coad and Palmer 2002), Lean Development (Charette 2002) and Adaptive Software Development (ASD, Highsmith 2002). Behind this family of methods are the agile values and agile principles as specified in the Agile Manifesto (http://www.agilemanifesto.org).

However, as Iivari and Iivari (2011) note, the Agile Manifesto, without considerable interpretation, cannot reasonably be used to identify agile methods and to distinguish them from other systems development

[1] http://www.leanmanufacturingconcepts.com/HistoryOfLeanManufacturing.htm

[2] http://finance.groups.yahoo.com/group/kanbandev/

methods. It is not necessarily self-evident what software development methods are agile, since they are not based on any clear common core idea except on the concept of agility, which itself is an ambiguous and multi-faceted concept (Conboy 2009). Iivari and Iivari (2011) also argue that one should distinguish agility as an a priori characterization of some software development methodologies and agility as an emergent feature that can be assessed only by hindsight. A characterization of the agile approach, based on the agile values and principles, is suggested by Iivari and Iivari (2011), as shown in Table 1.

Table 1. A Characterization of the Agile Approach (Iivari and Iivari 2011)

Goal	- To satisfy the customer through early and continuous delivery of software that is of value to the customer
Guiding principles	- Individuals and interactions are more significant in software development than processes and tools - It is more significant to respond to changing requirements than to follow a plan - The visibility assumption: Project visibility can be best achieved through the delivery of working code - The documentation assumption: Developing extensive (relatively complete) and consistent documentation and software models is counterproductive
Fundamental concepts	- Software as an emergent system: The best requirements, architectures, and design emerge
Principles of the development process	- Continuous or frequent delivery of working software - Focus on the software rather than other documentation - Welcoming changing requirements - Close collaboration between developers and customers

Lean Software Development

Viewed originally as just another agile method, lean software development is becoming a method category in itself rather than an instance of agile methods (Hibbs et al. 2009). has root in lean thinking which is a way of thinking that enables organizations to specify value, line up value-creating actions in the best sequence, conduct these activities without interruption whenever someone requests them, and perform them more and more effectively (Womack and Jones 1996). Five key concepts are highlighted in this definition:

- Value: Defined by the customer and it is paramount to have a clear understanding of what that is;
- Value stream: Identify every step in the process and categorise each step in terms of the value it adds;
- Flow: Maintain a continuous flow of the value-adding process;
- Pull: Customer orders pull product, ensuring nothing is built before it is needed;
- Perfection: Practice continuous improvement.

The primary focus of the lean approaches is on the identification and elimination of waste from the process. In the context of software development, waste can mean extra features, waiting, task switching, extra processes, partially done work, defects, etc. (Poppendieck and Poppendieck 2003, Hibbs et al. 2009).

Maintaining the core intent of lean, different lean principles for software development have been proposed. For example, Poppendieck and Poppendieck (2003) specify seven lean software development principles: Eliminate waste, Build quality in, Create knowledge, Defer commitment, Deliver fast, Respect people, and Optimise the whole. Based on the observation that software development processes is more a product development process rather than a manufacturing process, Reinertsen (2009) defines a set of principles of product development flow, including manage queues, reduce batch size, apply WIP (Work in Progress) constraints, etc.

The kanban approach is the most recent addition to the lean approaches. A kanban system is "a *production control system for just-in-time production and making full use of workers' capabilities*" (Sugimori et al., 1977). The core objective of the kanban system is to minimise the amount of WIP. Excess WIP is one form of waste from a lean perspective. Work should be "pulled" through the system as it is needed, as opposed to "pushing" it through. Only when a downstream process is ready and needs to do some more work does it pull work from an upstream process. Other properties of the kanban approach include visualising the workflow, making process policies explicit, and improve collaboratively using models and scientific method.

Agile vs. Lean Software Development

Traditionally, any conceptual comparison of agile and lean in the manufacturing and management literature would ultimately focus on the absence of value in lean thinking and the predominance of the same in the theory of agile (e.g., Burgess 2002, Katayama and Bennett 2004). However, as the concept of leanness has matured and evolved, value has prominently emerged as a key component (Lamming 1993, Hines and Taylor 2000, Hines et al. 2002), and so has closed the gap to some extent between lean and agile.

However, there are still several distinctions cited in these literature. As shown in Figure 1, Stratton and Warburton (2003) argue that an agile entity faces volatile demand compared to the more stable demand for standard products that a lean entity faces. Another distinction between a lean and agile organisation is that a lean one is cost efficient and productive, while an agile one learns fast if not initially cost efficient and productive (Stratton and Warburton 2003), even though the fact that leanness does not encourage learning would be hotly contested by some. While ultimate leanness eliminates all waste, agility requires waste to be eliminated, but only to the extent where its ability to respond to change is not hindered (Kusiak and He 1997), This does not remove the need to be economical, but only lowers its priority.

		Demand	
		Volatile	Stable
Product	Special	Agile	X
	Standard	X	Lean

Figure 1. Demand/Product Matrix for Agile and Lean Supply (Stratton and Warburton 2003)

When it comes to software development domain, the picture is more complex. Poppendieck and Poppendieck (2003) claim that lean is a philosophy and lean principles as they define provide the theory behind agile practices, and further expand the theoretical foundations of agile software development.

Others do not regard lean and agile at different levels, but consider them having different scopes and focuses (Ambler 2009, Smits 2007, Hibbs et al. 2009). Hibbs et al. (2009) believe that agile methods mostly concern themselves with the specific practice of developing software and the project management that surrounds that software development. They do not generally concern themselves with the surrounding business context in which the software development is taking place. Instead, lean principles can be applied to any scope, from the specific practice of developing software to the entire enterprise where software development is just one small part. The larger the scope, the larger the potential benefits. In addition, the primary focus of agile methods is on close customer collaboration and the rapid delivery of working software as early as possible whereas the primary focus of lean software development is on the elimination of waste in the context of what the customer values. Hibbs et al. (2009) also argue that agile methods have a fair number of prescribed practices, such as pair programming, standup meetings, whereas lean software development has no formal practices. It has a toolkit of recommended practices from which to choose. In terms of adoption style, they have the opinion that agile adoption generally takes a bottom-up approach whereas lean adoption is typically top-down.

It has also been argued that lean software development has the potential to resolve issues that agile methods have failed to address. Ambler (2009) argues that agile processes based on XP or Scrum can break down when one or more scaling factors are present (such as organizational wide agile adoption and transition or distributed agile development). Instead, a lean governance model based on enablement, collaboration, and motivation can enable agility at scale. Birkeland (2010) presents the experience of a software team shifting from a timebox-based development process (Scrum) to a flow-based process (pull-based, kanban), to show that flow-based process can better manage certain types of software development projects and activities, e.g. maintenance.

Middleton (2010) goes a step further and reflects on the issues with agile processes. He argues that agile processes, with time-boxed iterations, in essence still follow a push model. Instead in lean processes WIP (work-in-process) limits are used and work is "pulled" in when the team has capacity, therefore the team is not overloaded. He also argues that in agile processes benefits gained from retrospectives are largely anecdotal and not quantifiable. There is a risk that velocity estimates, number of features, or story points delivered are too subjective and easy to manipulate. In addition, even though agile processes have "inspect and adapt" in their retrospectives, the focus is more on the people rather than the work. Scrum "stand-up" directs attention to the people and what they did yesterday and what they are doing today. In contrast, lean processes rely on data for continual improvement. For example, lead time, which records total time from when a customer requested the work to when the finished work was received by the customer, is much harder to game. Meanwhile, a lean team enumerates the work, not the people. A lean team's "standup" focuses on their work and what the team was going to release. The data is used to help the team look up and down stream to enable innovation.

In brief, software practitioners have been arguing the differences between agile and lean processes. Preliminary studies and experience reports echo the opinions of the practitioners. However, there is no well-grounded and systemic study of the differences between agile and lean software development. Our study intends to fill in this knowledge gap.

Research Approach

As one way to answer the research question "what are the key differences between agile and lean processes", we are interested in understanding how the differences between agile and lean processes are perceived by wider software engineer communities that use the two approaches. To start with, we conduct a content analysis of a mailing list of the lean community. It is worth noting that much of the literature on the agile methods written by agile consultants and practitioners is not conceptually strong. The same may be the case of lean software development. Mailing list material of a lean community may be even more confusing and weak conceptually and hard to make sense. Therefore, a well-defined theoretical framework becomes more important to act as a sensitising and sense making device for this conceptually weak content.

Analysis of mailing lists is a commonly used research method (e.g., Lasker et al. 2005, Esquivel et al. 2006, Rigby and Hassan 2007). Content analysis has been defined in many different ways, but one of the most commonly cited is "a research technique for the objective, systematic, and quantitative description of the manifest content of communication" (Berelson 1952). There are four main types of content analysis ranging from simple to complex, namely syntactical, referential, propositional, and thematic (see Cooper and Schindler 2003 for a detailed description of these). This study will adopt different types at the different stages of data analysis.

It should be noted that although we propose that content analysis is a useful mechanism to explore the research question at a high-level, we also acknowledge that a number of issues existing in relation to content analysis techniques. A number of researchers have assessed the validity of content research, and have documented the various limitations and potential pitfalls of using the approach. The first possible pitfall occurs if the researcher is guilty of faulty definition and structure of content categories . The second pitfall is analysing content under categories that are not mutually exclusive . The final limitation of content analysis and especially simple word frequency counts is the failure to account for the context in which the words are used. Reliability of the data is a concern given that keywords found in different postings may not have been intended to convey the same meaning. This is particularly true given that both 'agile' and 'lean' are concepts shown to be multi-dimensional, vague and ambiguous (Conboy 2009). These issues need to be aware of to ensure the validity and reliability of the findings.

The chosen mailing list, "kanbandev", is one of the biggest and fastest growing discussion groups regarding lean software development. It was founded on Aug 20, 2007. And the subscribed members were 1587 on April 18, 2011 (which is still growing). These members are distributed in different countries across the globe. It is an active discussion forum for software practitioners who are practicing or interested in different aspects of lean software development. The discussions are not limited to kanban only.

The messages posted on the mailing list are 12,323 (by Mar 31st 2011). In order to sift and identify the discussed topics that are most relevant to our research question, we went through the following steps:

- Use the "Advanced" search function embedded in the web interface of the mailing list, and search the posts containing the relevant keywords. The keywords used are: "agile method" (which covers also "agile methods", "agile methodology" and "agile methodologies"), "agile process" (which covers also "agile" processes"), "XP", "extreme programming", and "Scrum". The assumption is that, since this is a lean mailing list, all the posts should be related to lean software development. Therefore those posts that contain any of the keywords above are about both agile and lean, to various degrees.
- For each keyword, group the returned search results into topics, again using the provided "Group by topics" function on the web interface of the mailing list. A discussion topic provides a natural boundary of a data set therefore is considered the unit of data analysis. It also preserves the complete context of a discussion, including who are the contributing members of the topic, the chronicle order, and the logical structure of the posts.
- To further reduce and identify the most relevant topics to be included in the in-depth content analysis, we examined the topics that have equal or more than 40 posts in the discussion body, and selected those that we deemed most relevant to the research question by quickly scanning the content of the posts. The assumption behind the choice of popular topics is that they reflect where the interests of the lean community lie.

At the end of Step 2, a number of 575 non-repetitive topics containing one or more search keywords are identified (after excluding apparently irrelevant topics, such as events announcements, advertisements, requests for research collaboration, etc.) As the result of Step 3, the topics were reduced to 37 (see the Appendix for the list of topics).

These 37 topics, relevant to the research question and well attended by the members, will be the main focuses for the in-depth data analysis phase. This analysis will be composed of three steps:

(1) Categorise topics into small groups: the intention is to group the topics according to their underlying descriptive themes, such as process improvement, standardization practices, combination of agile and lean, benefits/issues of transitioning from agile to lean, etc.. This step will allow the further data analysis more systematic and manageable. Even though the categories will emerge from this step of analysis, a reflection on what theme is apparently missing is equally important and informative.

(2) Analyse each individual topic in each group (since a topic is a unit of analysis): an open coding process defined in the Grounded Theory (Strauss and Corbin 1990) will be used to code the posts in each topic. The initial coding scheme will be built based on the constructed theoretical framework. The elements in the framework will be the seed codes. This step will result in an extended coding scheme.

(3) Contrast and merge the themes across the topics: the intention of this step is to search for common patterns across topics, in order to deepen the understanding and enhance the generalisability of the findings. Axial and selective coding processes defined in the Grounded Theory (Strauss and Corbin 1990) will be used to establish the common themes and patterns cross-cutting the different topics.

To manage the in-depth data analysis, we will use both Diigo (http://www.diigo.com), an online annotation tool, and NVivo, to analyse the 37 topics. Diigo will allow us to preserve the original thread structure of each topic, and to analyse the posts in their original context, in terms of the time they were posted, who were the authors of these posts, etc.. NVivo will allow a better manipulation of codes and identification of common patterns.

Future Work and Conclusion

The next stage of our study is to extend and refine the theoretical framework and conduct the in-depth data analysis. The existing framework developed by Stratton and Warburton (2003) within the supply context is fairly limited. Our literature review demonstrated that there are more, different dimensions on which the two approaches differ in a software development context (e.g., focus on customer, delivery speed, efficiency, focus on people, focus on work, measurement, etc.). Therefore, we will construct a theoretical framework of the differences between agile and lean that allows to account for the rich dimensions revealed by our literature review. More specifically, the comparison of agile and lean will be based on a set of common factors/variables, such as relative emphasis on customer interaction, efficiency and delivery speed or intervals. These factors/variables need clear definitions so that they can help strengthen the framework and promise a richer data analysis. A set of perceived key differences between agile and lean processes will be identified as the result of the analysis.

One limitation of our study lies in the data deduction process described in the Research Approach section. Even though it reduced the data to a manageable size (37 topics) effectively, it may also have excluded potentially useful data in the mailing list. A tradeoff has to be made between potential loss of useful data and the manageability of the data size. Another limitation of our research approach is that only one mailing list of the lean community is analysed. The generalisability of the findings to the whole lean community is limited as a consequence.

The findings of our study can facilitate a better understanding of both agile methods and lean approaches. It can also serve as a basis for the further research on the transition from agile and lean, to investigate the driving forces behind this trend. Another interesting study is to apply the same research approach to explore also the perspectives of the agile community on the differences between agile and lean, and compare the findings with those from the study of the lean community. We expect some really interesting results coming out of these investigations. Meanwhile, the adoption and adaptation of lean approaches to e.g. distributed development teams and different organisational cultures is also an interesting research avenue, which can be informed by the existing studies of the adoption and adaptation of agile methods. A good understanding of differences between agile and lean can facilitate this type of research.

Our study has also methodological implications. Mailing lists present a huge amount of unexplored yet potentially informative data. However there is no explicit, well-grounded description and discussion that help utilize them effectively in research. Our study is an example of how to tap into this valuable research resource.

Appendix

The list of topics in the "kanbandev" mailing list included in the in-depth data analysis

No.	Topic	Number of Posts
	2007	
1.	Is anybody doing it?	111
	2008	
2.	Kanban and Retrospectives	90
3.	Kanban, Flow and Cadence	82

	2009	
4.	Is there a lack of perceived schedule pressure in Kanban?	125
5.	Fellow travelers, it is time to free kanban from the shackles of its origin!	111
6.	Kanban vs Scrum	102
7.	"Standard Work" - Does it make sense for Software?	48
8.	how can Kanban improve technical practices	40
9.	Quality and pressure	106
10.	Kanban, Project Backlog, and Optional Scope Contracts/Timeboxing	79
11.	Comparing Apples with Oranges	70
12.	kanban is...	228
13.	pull and flow: what and where?	113
14.	Kanban and self organising teams	41
15.	In defense of WIP limits	48
16.	Comparing Apples with Apples?	41
17.	Lean/Kanban and Agile - follow on discussion to Alan's blog	113
18.	Commitment and Scrum/XP as a push system	73
19.	What are the hard questions?	44
20.	Lean - What has it given us?	47
	2010	
21.	Ken Schwaber: Waterfall, Lean/Kanban, and Scrum	143
22.	A wonderful success story	57
23.	Converting a Scrum team to Kanban - a case study	52
24.	Deming - I don't get it.	81
25.	Is there a Kanban Process?	67
26.	Re: [XP] Re: Is Kanban Agile?	53
27.	Why do we do Kanban?	48
28.	Waste as a red herring	46
29.	Thoughts on How Kanban Differs From Scrum	46
30.	A new take...	44
31.	Why we need Kanban	40
32.	Parallel work vs Don't start something new until you finish what you're doing	108
33.	The Principles of the Kanban Method	125
34.	Batching up releases... necessary or just patching over a symptom?	72
	2011	
35.	requesting help on myths of Kanban	75
36.	10 Principles of Lean Software Development	165
37.	Cobblestones On The Road to Perdition [ScrumAlliance]	150

Aknowledgment

This work is supported, in part, by Science Foundation Ireland grant 03/CE2/I303_1.

References

Ambler, S. W. 2009. "Scaling agile software development through lean governance." *SDG'09*, Vancouver, Canada, May 17, 2009.

Beck, K. 1999. *Extreme Programming Explained*, Addison Wesley, Reading, MA.

Berelson, B. 1952. *Content Analysis in Communication Research*, Glencoe, Ill: Free Press 1971 (first edition from 1952).

Birkeland, O. 2010. "From a Timebox Tangle to a More Flexible Flow," In *Proceedings of the 11th International Conference on Agile Processes in Software Engineering and Extreme Programming*, Springer Verlag, pp. 325-334.

Bruce, M., Daly, L., and Towers, N. 2004. "Lean or Agile: A Solution for Supply Chain Management in the Textiles and Clothing Industry," *International Journal of Operations and Production Management* (24:2), pp. 151-170.

Burgess, T. 2002. "Enhancing Value Stream Agility," *European Management Journal* (20:2), pp. 199-212.

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

Coad, P., and Palmer, S. 2002. *Feature-Driven Development*. Prentice Hall, Englewood Cliffs, NJ.

Cockburn, A. 2001. *Crystal Clear: A Human-Powered Software Development Methodology for Small Teams*, Addison-Wesley, Reading, MA.

Conboy, K. 2009. "Agility from first principles: Reconstructing the concept of agility in information systems development," *Information Systems Research* (20:3), pp. 329-354.

Cooper, D. R., and Schindler, P. 2003. *Business Research Methods*, McGraw Hill.

Dingsøyr, T., Dybå, T., and Abrahamsson, P. 2008. "A Preliminary Roadmap for Empirical Research on Agile Software Development," In *Proceedings of Agile 2008 Conference*, Toronto, 4-8 August.

Esquivel, A., Meric-Bernstam, F., and Bernstam, E. V. 2006. "Accuracy and self correction of information received from an internet breast cancer list: content analysis," *BMJ (Clinical research ed.)* (332: 7547), pp.939-942.

Hibbs, C., Jewett, S., and Sullivan, M. 2009. *The Art of Lean Software Development: A Practical and Incremental Approach*, 1st edition, O'Reilly Media, Inc., CA.

Highsmith, J. 2002. *Agile software development ecosystems*, Addison-Wesley, Boston.

Hines, P., and Taylor, D. 2000. *Going Lean: A Guide for Implementation*. Cardiff: Lean Enterprise Research Centre, Cardiff Business School.

Hines, P., Silvi, R., and Bartolini, M. 2002. "Demand Chain Management: An Integrative Approach in Automotive Retailing," *Journal of Operations Management* (20:3), pp. 707-28.

Iivari, J., and Iivari, N. 2011. The relationship between organizational culture and the deployment of agile methods. *Information and Software Technology (*53:5), pp.509-520.

Katayama, H., and Bennet, D. 1999. Agility, Adaptability and Leanness: A Comparison of Concepts and a Study of Practice. *International Journal of Production Economics* (62:1/2), pp. 43-51.

Kusiak, A., and He, D. 1997. "Design for Agile Assembly: An Operational Perspective," *International Journal of Production Research* (35:1), pp. 157-178.

Lamming, R. 1993. *Beyond Partnership: Strategies for Innovation and Lean Supply*. London: Prentice Hall.

Lasker, J., Sogolow, E., and Sharim, R. 2005. "The role of an online community for people with a rare disease: content analysis of messages posted on a primary biliary cirrhosis mailinglist," *Journal of medical Internet research* (7:1), e10.

Mason-Jones, R., Naylor, J., and Towill, D. 2000. "Engineering of the Leagile Supply Chain," *International Journal of Agile Management Systems* (2:1), pp. 54-61.

Middleton, P. 2010. "Lean Software Management: BBC Worldwide Case Study", In *Proceedings of Lean Software & Systems Conference*, Atlanta, pp. 23-44.

Naylor, J., Naim, M., and Berry, D. 1999. "Leagility: Integrating the Lean and Agile Manufacturing Paradigm in the Total Supply Chain," *Engineering Costs and Production Economics* (62:1), pp. 107-

118.

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

Reinertsen, D. G. 2009. *The Principles of Product Development Flow: Second Generation Lean Product Development.* Celeritas Publishing.

Rigby, P., and Hassan, A. 2007. What Can OSS Mailing Lists Tell Us? A Preliminary Psychometric Text Analysis of the Apache Developer Mailing List. *MSR '07 Proceedings of the Fourth International Workshop on Mining Software Repositories,* Washington, DC, 2007.

Roberts, C. E. 1997. *Text Analysis for the Social Sciences: Methods for Drawing Statistical Inferences from Texts and Transcripts*, Mahwah, NJ, Lawrence Erlbaum.

Rutherford, K., Shannon, P., Judson, C., and Kidd, N. 2010. "From Chaos to Kanban, via Scrum," *Proceedings of the 11th International Conference on Agile Software Development, XP2010*, Trondheim, Norway: Springer Verlag, pp. 344-352.

Schwaber, K., and Beedle, A. 2002. *Agile Software Development with SCRUM*, Prentice-Hall, Upper Saddle River, NJ.

Smits, H. 2007. "The Impact of Scaling on Planning Activities in an Agile Software Development Center," In *Proceedings of the 40th Hawaii International Conference on System Sciences*, Waikoloa, Big Island, Hawaii, 3-6 January 2007.

Stratton, R., and Warburton, R. 2003. "The Strategic Integration of Agile and Lean Supply," *International Journal of Production Economics* (85:1), pp. 183-188.

Strauss, A.C., and Corbin, J.M. 1990. *Basics of Qualitative Research: Grounded Theory Procedures and Techniques*, Sage Publications, Inc.

Sugimori, Y., Kusunoki, K., Cho, F., Uchikawa, S. 1977. Toyota production system and Kanban system Materialization of just-in-time and respect-for-human system. *International Journal of Production Research (*15), pp. 553 – 564.

VersionOne, 2010. 5th Annual State of Agile Development Survey, *[Online]. Available: http://www.versionone.com/pdf/2010_State_of_Agile_Development_Survey_Results.pdf* [Accessed 01/09/2011]

Weber, R. 1990. *Basic Content Analysis*, Newbury Park, CA. , Sage.

Willeke, E. R. 2009. "The Inkubook Experience: A Tale of Five Processes," In *Proceedings of Agile 2009 Conference*, Chicago, USA, pp. 24-28.

Womack, J. P., Jones, D. T. 1996. *Lean Thinking : Banish Waste and Create Wealth in Your Corporation*. Simon & Schuster.

Womack, J., Jones, D., and Roos, D. 1990. *The Machine That Changed the World*. NY, Rawson Associates.