What Does Research Say About Agile and Architecture?

Hongyu Pei Breivold
Industrial Software Systems
ABB Corporate Research
721 78 Västerås, Sweden
Hongyu.pei-breivold@se.abb.com

Daniel Sundmark, Peter Wallin and Stig Larsson
Mälardalen University
Box 883, 721 23 Västerås, Sweden
{daniel.sundmark, peter.wallin, stig.larsson}@mdh.se

***Abstract*— Agile has been used to refer to a software development paradigm that emphasizes rapid and flexible development. In the meanwhile, we have through our practical experiences in scaling up agile methods, noticed that architecture plays an important role. Due to the inter-relationship between agile methods and architecture, as well as divergent perceptions on their correlation stated in numerous sources, we are motivated to find out how these perceptions are supported by findings in the research community in general and in empirical studies in particular. To fully benefit from agile practices and architectural disciplines, we need empirical data on the perceived and experienced impacts of introducing agile methods to existing software development process, as well as correlations between agile and architecture. In this paper, we survey the research literature for statements made regarding the relationship between agile development and software architecture. The main findings are that there is a lack of scientific support for many of the claims that are concerned with agile and architecture, and more empirical studies are needed to fully reveal the benefits and drawbacks implied by an agile software development method.**

Keywords-Agile; Agile methodology; Architecture

I. INTRODUCTION

To cope with the fact that development and maintenance of software is characterized by constant evolution, a number of methodologies and practices have been developed to embrace change. These methodologies are designated as being "agile", referring to a software development paradigm that emphasizes rapid and flexible development. Through our involvement in scaling up agile methods (e.g., in the FLEXI project [42]), we noticed that architecture is an important topic, especially when development is distributed over several sites, and/or when the development is based on legacy systems. As an example, interfaces, being an important component of the architecture, are vital to create an understanding of the interaction between the different parts of the system. In addition, we have through our earlier research seen that changes in architecture influence processes, and vice versa [1, 2], prompting organizations to consider changes in both areas when one of them is altered. We have also observed a discrepancy between the literature on software architecture (e.g., [3] and [4]) and agile methods (e.g., [5] and [6]). An extreme way of describing this difference is that either one has to make a number of key decisions that will decide the characteristics of a system before building it, or one has to work with the basic structure throughout the life-time of the system ensuring that only supported features are accommodated.

Although agile software development methodologies have received great positive attention in recent years, there exist divergent perceptions on the correlation between agile methods and architecture. While considered to be able to improve performance in development projects, agile methods have also been criticized by some practitioners and academics for their lack of architectural focus [7, 8]. As stated in [9], "agile methodologies advocate many good engineering practices, although some practices may have an extreme implementation that is controversial and counterproductive outside a narrow domain". For instance, as change is inevitable and planning for future functions is a waste of effort, eXtreme Programming (XP) advocates doing extra work to eliminate architectural features that are not related to the system's current version [5]. It could be argued whether such a practice jeopardizes the architectural support for customers' potential requirements. Another fear is that an over-focus on early results in large systems can lead to major rework when the initial architecture does not scale up [10]. Agile proponents generally suggest that these issues can be handled using agile practices and techniques, e.g., test-driven development (TDD) and refactoring. Critics however claim that such practices are too lightweight and do not meet the architectural needs to a sufficient degree. Accordingly, an interesting research question that is posed becomes:

RQ1: Is architecture sufficiently emphasized in agile methods?

By looking into agile practices, e.g., TDD and refactoring in particular, we explore into a related research question:

RQ2: Do agile practices improve software architecture?

In this article, we survey the research literature for statements and claims made regarding the relationship between agile development and software architecture. Our main contribution is a summary of research findings on the relationship between agile development and software architecture. Our main observation is that there is a lack of scientific support for many of the claims made by the agile community. Many claims are solely based on expert opinion, and often, the adoption of an agile approach is tested in student projects. Even though there are some articles describing the adoption of agile methods in industrial cases, these often focus on specific agile practices, e.g., TDD, pair programming, etc. More empirical research is therefore essential to fully exploit the relationship between agile development and software architecture.

II. RESEARCH METHOD

This study was undertaken as a systematic literature review during March 2009, based on the original guidelines as proposed by Kitchenham [11]. The study includes several stages: (i) the identification of inclusion and exclusion criteria for primary studies; (ii) the search process for relevant studies; and (iii) the extraction and synthesis of data from the selected primary studies.

A. Inclusion and Exclusion Criteria

We consider full papers in English from peer-reviewed journals, conferences and workshops. We exclude studies that do not relate to software engineering, software architecture and agile methodologies. We also exclude prefaces, articles in the controversial corner of journals, editorials, summaries of tutorials, panels and poster sessions.

B. Search Process

We searched in scientific databases, i.e., ACM Digital Library, Compendex, IEEE Xplore, ScienceDirect – Elsevier, SpringerLink, Wiley InterScience and ISI Web of Science. The search terms that were used to find relevant studies are presented in Table 1. All these search terms were combined by using the Boolean OR operator.

Table 1. Summary of Search Terms Used in the Review

	Search Terms
S1	*"software architecture" AND agile*
S2	*"software architecture" AND "extreme programming"*
S3	*"software architecture" AND XP*
S4	*"software architecture" AND lean*
S5	*"software architecture" AND DSDM*
S6	*"software architecture" AND "dynamic systems development method"*
S7	*"software architecture" AND FDD*
S8	*"software architecture" AND "feature driven development"*
S9	*"software architecture" AND scrum*
S10	*"software architecture" AND crystal*
S11	*"software architecture" AND ASD*
S12	*"software architecture" AND "adaptive software development"*

After an initial search in electronic databases, we did an additional reference scanning and analysis by going through the references from the included papers, in order to find out if we were able to detect any studies that had been erroneously omitted or missed, thus to guarantee a representative set of studies. The study selection process was performed through several steps:

(i) Search in databases and conference proceedings to identify relevant studies; 842 studies (after removing duplicates) were identified in this step.

(ii) Exclusion of studies based on the formal exclusion criteria (e.g., exclusion of non-peer-reviewed papers); 409 studies were excluded in this step.

(iii) Exclusion of studies based on titles and abstracts; 238 studies were excluded in this step.

(iv) Exclusion of studies based on full text. 159 studies were further excluded.

In the paper selection process, there were always at least two researchers involved in order to decide whether to include or exclude a paper. A paper is excluded if both researchers consider it irrelevant. When there were any discrepancies in whether to include or exclude a paper, discussions were then initiated in order to reach an agreement. In the end, we had in total 36 studies that were included in the systematic review for synthesizing the relation between agile and architecture.

C. Data Extraction and Synthesis

In the data extraction process, we focused on statements made on how architecture and agile development influence each other. The primary data for extraction is concerned with the claims that are made regarding agile and architecture as well as the rationale and validation for these claims. The data extracted from each study include the source and full reference, objectives and focus of the study, research method descriptions, data collection and analysis, findings and conclusions. The data synthesis process included identifying the main concepts from each study and analyzing how they relate to our research questions. Accordingly, the statements made on agile and architecture in the included 36 papers were categorized into different themes based on their similarities in terms of contents. Similar to the paper selection process, at least two researchers were involved in the data synthesis process to ensure an unbiased categorization of these statements.

III. SURVEY FINDINGS

After examining and synthesizing the data from the included studies, we present a summary of the findings that address the two research questions.

A. RQ1: Is architecture sufficiently emphasized in agile methods?

Two aspects are addressed in this section: (i) claims on the interplay of agile and architecture; and (ii) empirical studies concerned with agile and architecture interplay.

1) Agile and architecture: As agile methods in general describe very sparsely how to handle architecture, several studies have advocated the need to synthesize and extend

agile development with methods supporting architecture. Nord and Tomayko [12] argue that software architecture-centric methods can enhance XP practices and add value to agile methods by emphasizing quality attributes and their role in shaping the architecture's design through scenarios. In addition, as agile approaches emphasize face-to-face communication to convey information, the risk of making irrecoverable architectural mistakes because of unrecognized shortfalls in its tacit knowledge can be complemented with usage of common concepts, e.g., quality attributes, architectural tactics and views-based architecture documentation approach from architecture-centric methods. According to Boehm, agile approaches have home-ground areas that they are appropriate for [10]. The difference in the architecture area is that agile methods are designed for current requirements, whereas plan-driven and architecture-centric methods are designed for current and foreseeable requirements. Accordingly, Boehm states that hybrid approaches which combine both agile and plan-driven methods are feasible and necessary for projects that have a mixture of agile and plan-driven characteristics.

It has been reported that synthesizing agile practices with architectural methods requires that care is taken so as not to jeopardize agile philosophies. Specifically, Sharifloo et al. [13] cast doubts on practical integration of these architecture-centric methods, and argues for the need of using architectural methods in XP without disobeying the atmosphere of the XP development team and XP's values. A successful integration is described by Clements et al. [14], who present an example in reconciling the traditional and agile approaches capturing software architectural information in a manner consistent with agile philosophies.

Several researchers address the topic of augmenting specific software engineering practices and agile methods. Abrahamsson et al. [15] describe the experiences of augmenting agile methods by systematically using an architectural practice called Architectural Lines to capture "current architectural knowledge about the patterns and solutions that have proven to be useful". The combination of agile methods and the new architectural practice led in the case study to "increased progress visibility, early identification and solving of technical problems, shared responsibility, efficient information sharing, high process practice coherence, low defect density in released products, and constant development rhythm". Several other studies suggest that existing agile methodologies can be integrated in other software engineering paradigms, e.g., model-driven development [16, 17], product line engineering [18, 19], and service-oriented technologies [20].

Some examined studies describe the values of using agile methods and the necessity of applying agile thinking to existing architectural activities. For instance, Davide et al. [21] advocate the need to use lightweight and agile methods to reduce the effort both in creating and consuming architectural knowledge. Moreover, Hadar and Silberman [22] propose an Agile Architecture methodology which combines quick feedback by delivering in short incremental releases with the use of an architectural roadmap.

In summary, the current research in the area of agile methods and architecture-centric methods claims that there is a need to combine these two areas to ensure both quick response to changing market needs, and long-term survival of product assets. This synthesis should be done with care to work well together with the existing good developmental practices. However, this claim is insufficiently supported by empirical work, and is mostly backed up by expert opinion. Moreover, the cited articles in this subsection are motivated by an assumption that agile methods insufficiently meet architectural demands. This assumption is further analyzed in the next subsection. On the other hand, the observations regarding architecture and agile are most likely based on practitioners' experiences, and has been observed as an area where additional practices are needed. Based on this, we think that there would be value in additional research to understand the influence between architecture and agile before additional practices are proposed.

2) Empirical studies in agile and architecture: A commonly stated assumption and motivation for research focusing on agile and architecture, is that the lack of emphasis on architecture in agile practices leads to architectural problems [8, 13, 19, 22-30]. However, when searching for empirical findings that actually support this assumption, we find that such results are sparse. On the other hand, equally few empirical results support the opposite position: that agile development supports architectural development better than traditional software development methods. The empirical studies conceiving that architecture is insufficiently emphasized in agile methods include a longitudinal case study by Hanssen and Faegri [25] on agile development in a small software company. In the study, developers state that "the continuous focus on direct customer value weakens the focus on engineering handcraft such as thorough general design". The concern is that the focus on short-term goals may lead to development shortcuts, resulting in a degradation of the architecture. Moreover, based on a study of nine US internet software development organizations, Ramesh et al. [31] observe a tendency for developers to "move towards more traditional approaches to software development as their products and markets matured and the complexity of the development grows", indicating that agile methods are insufficient in supporting complex architecture. It should be noted that the study by Hanssen and Faegri actually reports from a case where the developers involved felt that the agile method at hand was unable to cope with architectural integrity, whereas observations in the study by Ramesh et al. [31] can be interpreted as a symptom of this inability. However, there may be other interpretations of these observations.

The empirical results that advocate that architecture is sufficiently emphasized in agile methods include observations from a study by Wellington [32], where two student teams were assigned to work in a software

development project. One team worked according to a traditional life cycle (TLC) development process, whereas the other team used XP. According to the study, an over-complicated up-front architecture of the TLC team resulted in a significantly lower quality than that delivered by the XP team, who "took advantage of multiple iterations and automated tests to refactor their code on multiple occasions". A similar observation is made by Ambler in [33]. Based on experiences from introducing agile in two Internet startups, the author concludes that "big up-front design isn't required". The only architectural work needed is a "few afternoons" to formulate an initial high-level design.

In summary, it is hard to conclude anything based on the sparse empirical results supporting or contradicting the assumption that architecture is insufficiently emphasized in agile methods. The studies that do exist are small, diverging and, in some cases, performed in an artificial setting. This is noteworthy considering the large number of articles using this assumption, or subtle variants thereof, as a motivation for research contributions acting to solve the issue.

B. *RQ2: Do agile practices improve software architecture?*

This section presents research studies that are concerned with the correlation between architecture/TDD and architecture/refactoring.

1) TDD and architecture: TDD is a technique that encourages simple designs [34]. As XP originator Kent Beck asserts, "Test-first code tends to be more cohesive and less coupled than code in which testing isn't a part of the intimate coding cycle" [35]. However, George and Williams [24] argue that the lack of upfront design, as well as the emphasis on implementation rather than logical structure in TDD might be a concern for software practitioners, and that these issues call for the need of empirical analysis of TDD.

In fact, the effect of TDD on software architecture, and on software quality in general, is an area where statements are rooted in empirical findings to a greater extent than the other findings in this paper. Moreover, contrary to an existing assumption that agile methods and architecture do not mix, most findings actually support the view that TDD acts to strengthen architectural development.

George and Williams [24], and Janzen and Saiedian [36] observe a tendency for TDD programmers to write simpler, less complex software. In the case of [36], this conclusion is based on observations of lower complexity metrics in code produced by TDD programmers in a set of quasi-controlled experiments and a case study. In the case of [24], it is based on results from a post-experimental survey. The authors state that "TDD proponents argue that reduced coupling occurs because the practice guides them to the building of objects that are actually needed (to pass test cases based on the requirements) rather than building objects that are thought to be needed". It should be noted that [36] were unable to confirm a reduced coupling in TDD-developed code.

An improved design and modularity in code produced by TDD developers has been observed in a case study by Cordeiro et al. [37]. This conclusion is also supported by the survey results reported by George and Williams [24]. There is however no detailed description of what actually is meant by the term "improved design". As an indirect effect of improved modularity, a more resource-efficient testing is reported by Nelson and Kim [30].

A clear indication from empirical results regarding TDD concerns increased external code quality (e.g., reduced defect density), which, per se, is not a direct property of software architecture. However, if TDD systematically would produce poorer software architecture than more traditionally developed code, arguably, we would not consistently observe an increased external code quality in TDD-developed software. A defect reduction of 40% in TDD-developed code compared to traditionally developed baseline code has been observed in a case study at IBM performed by Williams et al. [38]. Moreover, George and Williams report that approximately 18% more functional test cases are passed than in the control group pairs [24], when TDD is compared to software developed in a waterfall-like manner. In the same study, notably high structural code coverage measures were observed. The increased code quality effect of TDD has also been described in a recent study by Marchenko et al. [39].

For fairness sake, it should be noted that TDD comes with a price; problems reported with introducing TDD include an increased time to develop test cases [24], transitioning problems for individual developers [24], and loss of design and architecture decision traceability [17]. It has also been stated that TDD can only be successfully introduced when there is a good understanding of the code base, and the code base is in good shape [40].

As a summary, even though there is no massive body of evidence, the perception that TDD has a positive effect on architectural quality has the highest scientific validity from empirical study perspective. It is also interesting to note that most statements in favor of TDD when it comes to developing software with high-quality architecture are based on empirical evidence, whereas most authors speaking against TDD in this matter solely base their statements on expert opinion, i.e., non-justified or ad hoc statements instead of based on evidence.

2) Refactoring and architecture: In a software life-cycle perspective, refactoring will presumably be inevitable at some point, regardless of development method used. However, as Van Gurp et al. [41] conclude, the iterative nature of agile approaches is likely to increase the need for an early refactoring of the architecture. In their study, Van Gurp et al. categorize two extremes of iterative development, i.e., the *minimal effort strategy*, and the *optimal design strategy*. The former uses a more agile approach where only the next iteration is considered and changes are kept to a minimum. This approach will cause an architectural drift and erosion that will drive a refactoring of the architecture since the integrity is violated. The optimal strategy, on the other hand, hypothesizes that all necessary changes to the software are made for each new set of requirements. "However, even the optimal strategy does not lead to an optimal design. It just delays inevitable problems

like design erosion and architectural drift." [41]. In [27], Juric identifies similar experiences with XP, stating that the early converting of unit tests to production code will require more frequent and early refactoring. Further, George [24] argues that one of the shortcomings of TDD is the reliance on refactoring as a mean to reduce or maintain complexity.

In summary, synthesis of the above statements leads to the conclusion that refactoring is inevitable, but using architectural principles and at the same time understanding the requirements posted by the architecture can limit both its magnitude and frequency. Nonetheless, these statements are based on expert opinion rather than empirical results, and this is a field where future research is encouraged.

IV. Discussion

Based on somewhat contradictory findings it is hard to conclude neither that agile and architecture is like oil and water, nor that the two are the perfect marriage.

A. Related Non-Scientific Work

In addition to the scientific contributions on agile and architecture, there is an ongoing discussion about this topic in the software engineering community. One example is a reoccurring workshop in conjunction to the XP-series of conferences, supported by a wiki [43]. Moreover, researchers and practitioners exchange experiences and ideas in other wikis such as in the Agile 2.0 section of the OBJECTWARE Open Community [44], and in social networks, such as the Saturn group in LinkedIn. A lot of information regarding architecture and agile is available from work-in-progress workshops, seminars, company presentations, and websites (including blogs). However, as most of this information is based on non peer-reviewed expert opinion, we have not considered it in the scope of this paper. This limits the amount of information that we have used for the conclusions, but ensures a certain level of quality.

B. Validity Threats

In addition to the fact that software engineering is a young and fairly difficult area for empirical research (it is, e.g., hard to establish causality due to the many different factors that may affect outcome), there are well-founded claims that study design, and methodological rigor could be improved in this area [11]. Hence, it may come as no surprise that the vast majority of statements made about the effect on architecture by agile development (and vice versa) hold a limited scientific validity. While some of the collected data is based on empirical evidence, the largest part of the data is based solely on expert opinion. Specifically, out of the 130 statements collected in our study, 39 are based on empirical evidence (33 are based on case studies and 6 on experiments). The remaining 91 statements are based on expert opinion. In addition, only 15 of the 34 selected publications based their reasoning, or parts thereof, on any type of explicit empirical evidence.

A threat to construct validity is the use of not clearly defined terms, e.g., *agile* and *architecture*. We dealt with this threat by making sure that all the researchers participating in this review had the same definition in case of unclear terms. In some cases it was hard to know how the authors of the reviewed papers defined for example agile or architecture. By ensuring that two researchers were part of the exclusion of papers as well as the data analysis part we could discuss possible interpretations and agree on it.

To ensure that we actually used the correct search strings to answer our research questions we did a protocol including a description of the study, search strings that we intended to use and the research questions we sought to answer. This protocol was sent to three senior researchers for feedback and the protocol was modified based on their comments.

Regarding external validity, it is hard to make any general claims about architecture and agile. Basically, the lack of empirical studies makes it hard or even impossible to generalize any of the conclusions.

V. Summary and Future Research Directions

As noted above, the research performed in the intersection of agile and architecture is scattered, often performed in small settings, seldom based on agreed-upon supporting metrics, and sometimes tends to draw conclusions outside the scope of the study. Hence, we think that larger studies, based on defined metrics, performed in the industrial domain, are necessary in order to increase our understanding of how agile and architecture interrelate. This understanding would be beneficial in determining when agile methods are suitable, when they need to be complemented, and when other development method are more suitable.

Even though the included publications in our survey vary in focus, domain, study type and study quality, we have made no such considerations when synthesizing our claims, apart from the distinction made between statements based on empirical evidence, and those based on expert opinion. A more detailed analysis considering the above factors might provide more clear results, but at the present time, we consider the accumulated body of empirical evidence too small for such an analysis, which remains to be our future research when there is a wide range of empirical data.

References

[1] S. Larsson, A. Wall, and P. Wallin, "Assessing the influence on processes when evolving the software architecture," in Ninth international workshop on Principles of software evolution: in conjunction with the 6th ESEC/FSE joint meeting Dubrovnik, Croatia: ACM, 2007.

[2] R. Land, J. Carlson, S. Larsson, and I. Crnković, "Towards Guidelines for a Development Process for Component-Based Embedded Systems," in Computational Science and Its Applications – ICCSA 2009, 2009, pp. 43-58.

[3] L. Bass, P. Clements, and R. Kazman, Software Architecutre in Practice, 2nd ed. Boston, MA: Addison Wesley, 2003.

[4] D. Garlan, "Software architecture: a roadmap," in Proceedings of the Conference on The Future of Software Engineering Limerick, Ireland: ACM, 2000.

[5] K. Beck, Extreme Programming Explained: Embrace Change: Addison-Wesley Professional, 1999.

[6] K. Schwaber and M. Beedle, Agile Software Development with Scrum: Prentice Hall PTR, 2001.

[7] T. Dybå and T. Dingsøyr, "Empirical studies of agile software development: A systematic review," Information and Software Technology, 2008.

[8] L. Hochstein and M. Lindvall, "Combating architectural degeneration: a survey," Information and Software Technology, vol. 47, pp. 643-656, 2005.

[9] M. C. Paulk, "Agile methodologies and process discipline," Crosstalk, pp. 15-18, 2002.

[10] B. Boehm, "Get ready for agile methods, with care," Computer, vol. 35, pp. 64-69, 2002.

[11] B. Kitchenham, "Procedures for performing systematic reviews," Keele University TR/SE-0401/NICTA Technical Report 0400011T, vol. 1, 2004.

[12] R. L. Nord and J. E. Tomayko, "Software architecture-centric methods and agile development," IEEE Software, vol. 23, pp. 47-53, 2006.

[13] A. A. Sharifloo, A. S. Saffarian, and F. Shams, "Embedding Architectural Practices into Extreme Programming," in Software Engineering, 2008. ASWEC 2008. 19th Australian Conference on, 2008, pp. 310-319.

[14] P. Clements, J. Ivers, R. Little, R. Nord, J. Stafford, and I. Carnegie-Mellon Univ Pittsburgh Pa Software Engineering, Documenting Software Architectures in an Agile World: Carnegie Mellon University, Software Engineering Institute, 2003.

[15] P. Abrahamsson, A. Hanhineva, H. Hulkko, T. Ihme, J. Jäälinoja, M. Korkala, J. Koskela, P. Kyllönen, and O. Salo, "Mobile-D: an agile approach for mobile application development," in Companion to the 19th annual ACM SIGPLAN conference on Object-oriented programming systems, languages, and applications, Vancouver, BC, CANADA, 2004, pp. 174-175.

[16] F. Chitforoush, M. Yazdandoost, and R. Ramsin, "Methodology support for the model driven architecture," in 14th Asia-Pacific Software Engineering Conference (APSEC), Los Alamitos, CA 90720-1314, United States, 2007, pp. 454-461.

[17] D. E. Perry and P. S. Grisham, "Architecture and design intent in component & COTS based systems," in Commercial-off-the-Shelf (COTS)-Based Software Systems, 2006. Fifth International Conference on, 2006, p. 10 pp.

[18] M. Karam, S. Dascalu, H. Safa, R. Santina, and Z. Koteich, "A product-line architecture for web service-based visual composition of web applications," Journal of Systems and Software, vol. 81, pp. 855-867, 2008.

[19] M. Raatikainen, K. Rautiainen, V. Myllärniemi, and T. Männistö;, "Integrating product family modeling with development management in agile methods," in Proceedings of the 1st international workshop on Software development governance, Leipzig, Germany, 2008, pp. 17-20.

[20] I. H. Krueger, M. Meisinger, M. Menarini, and S. Pasco, "Rapid Systems of Systems Integration Â¿ Combining an Architecture-Centric Approach with Enterprise Service Bus Infrastructure," in Information Reuse and Integration, 2006 IEEE International Conference on, 2006, pp. 51-56.

[21] F. Davide, C. Rafael, and C. Giovanni, "A value-based approach for documenting design decisions rationale: a replicated experiment," in Proceedings of the 3rd international workshop on Sharing and reusing architectural knowledge Leipzig, Germany: ACM, 2008.

[22] E. Hadar and G. M. Silberman, "Agile architecture methodology: long term strategy interleaved with short term tactics," in Companion to the 23rd ACM SIGPLAN conference on Object-oriented programming systems languages and applications Nashville, TN, USA: ACM, 2008.

[23] D. Assmann and T. Punter, "Towards partnership in software subcontracting," Computers in Industry, vol. 54, pp. 137-150, 2004.

[24] B. George and L. Williams, "A structured experiment of test-driven development," Information and Software Technology, vol. 46, pp. 337-342, 2004.

[25] G. K. Hanssen and T. E. Faegri, "Agile customer engagement: a longitudinal qualitative case study " in Proceedings of the 2006 ACM/IEEE international symposium on Empirical software engineering, Rio de Janeiro, Brazil, 2006, pp. 164-173.

[26] G. K. Hanssen and T. E. Fægri, "Process fusion: An industrial case study on agile software product line engineering," Journal of Systems and Software, vol. 81, pp. 843-854, 2008.

[27] R. Juric, "Extreme programming and its development practices," in Information Technology Interfaces, 2000. ITI 2000. Proceedings of the 22nd International Conference on, 2000, pp. 97-104.

[28] H. Obendorf and M. Finck, "Scenario-based usability engineering techniques in agile development processes," in CHI '08 extended abstracts on Human factors in computing systems, Florence, Italy, 2008, pp. 2159-2166.

[29] K. R. Schougaard, K. M. Hansen, and H. B. Christensen, "SA@Work - A Field Study of Software Architecture and Software Quality at Work," Apsec 2008:15th Asia-Pacific Software Engineering Conference, Proceedings, pp. 411-418, 2008.

[30] C. Nelson and J. S. Kim, "Integration of software engineering techniques through the use of architecture, process, and people management: An experience report," Rapid Integration of Software Engineering Techniques, vol. 3475, pp. 1-10, 2005.

[31] B. Ramesh, J. Pries-Heje, and R. Baskerville, "Internet Software Engineering: A Different Class of Processes," Annals of Software Engineering, vol. 14, pp. 169-195, 2002.

[32] C. A. Wellington, T. Briggs, and C. D. Girard, "Examining team cohesion as an effect of software engineering methodology," in Proceedings of the 2005 workshop on Human and social factors of software engineering, St. Louis, Missouri, 2005, pp. 1-5.

[33] S. W. Ambler, "Lessons in agility from Internet-based development," Software, IEEE, vol. 19, pp. 66-73, 2002.

[34] Beck, Test Driven Development: By Example: Addison-Wesley Longman Publishing Co., Inc., 2002.

[35] K. Beck, "Aim, Fire," IEEE Softw., vol. 18, pp. 87-89, 2001.

[36] D. S. Janzen and H. Saiedian, "Does Test-Driven Development Really Improve Software Design Quality?," Software, IEEE, vol. 25, pp. 77-84, 2008.

[37] L. Cordeiro, C. Mar, E. Valentin, F. Cruz, D. Patrick, R. Barreto, and V. Lucena, "An agile development methodology applied to embedded control software under stringent hardware constraints," SIGSOFT Softw. Eng. Notes, vol. 33, pp. 1-10, 2008.

[38] L. Williams, E. M. Maximilien, and M. Vouk, "Test-driven development as a defect-reduction practice," in Software Reliability Engineering, 2003. ISSRE 2003. 14th International Symposium on, 2003, pp. 34-45.

[39] A. Marchenko, P. Abrahamsson, and T. Ihme, "Long-Term Effects of Test-Driven Development A Case Study," in Agile Processes in Software Engineering and Extreme Programming, 2009, pp. 13-22.

[40] A. van Deursen, "Program comprehension risks and opportunities in extreme programming," in Reverse Engineering, 2001. Proceedings. Eighth Working Conference on, 2001, pp. 176-185.

[41] J. van Gurp and J. Bosch, "Design erosion: problems and causes," Journal of Systems and Software, vol. 61, pp. 105-119, 2002.

[42] http://www.flexi-itea2.org/index.php (visited April 2010)

[43] http://www.lmsa-community.org/wikis/index.php/Architecture-Centric_Methods_and_Agile_Approaches (visited April 2010)

[44] http://wiki.community.objectware.no/display/smidigtonull/Agile+and+Software+Architecture (visited April 2010)