December 2007

An Empirical Study Examining the Usage and Perceived Importance of XP Practices

Ann Fruhling
UNO

Jessica Zhang
UNO

An Empirical Study Examining the Usage and Perceived Importance of XP Practices

Ann Fruhling
College of Information Science and Technology
University of Nebraska at Omaha
afruhling@mail.unomaha.edu

Daozhen Zhang
College of Information Science and Technology
University of Nebraska at Omaha
dzhang@mail.unomaha.edu

Abstract

Extreme Programming (XP) is a well known agile software development methodology which is ideal for projects featured as highly unpredictable in tasks with limited resources. The continuous discussion on the usage and importance of each XP practice lead us to explore what are the most important XP practices to be applied in certain projects. This study examined the actual usage amount and perceived importance of each XP practice by means of a cross-sectional anonymous survey conducted in local organizations which have implemented XP in their projects. Results indicate that Continuous Integration and Collective Ownership as the most important. Collective Ownership, Continuous Integration, Pair Programming, Planning Game and Sustainable Pace are used the most. Both practitioners and researchers can build upon these findings.

Keywords – eXtreme Programming, Agile Methods, Software Development

Introduction

Businesses today compete in an environment where customer needs are constantly evolving due to new dynamic technology capabilities. Consequently, businesses need to be more agile, especially when developing new information systems and software. The implied precondition of the past where customers knew exactly what they wanted at the beginning of the information system development process is less and less the case. Organizations that satisfy customer needs, deliver required software quickly, and are responsive to the business conditions often have a competitive edge.

Because of these dynamic business conditions, several light-weight software development processes have emerged. Light-weight processes aim to be more adaptive, responsive and productive. They also aspire to bridge the gap between the fast-paced business environment and the time-intensive system development process. One well known agile software development process, eXtreme Programming (XP), embraces highly unpredictable tasks (Beck 2004). In fact, the XP approach welcomes change, emphasizes customer interaction, and seeks to overcome the limitations of plan-driven software development methodologies (Lippert et al. 2003).

Therefore, many organizations have decided to introduce and implement agile methods, especially eXtreme Programming, in their system development projects. However, there is on-going debate whether all or most of the XP practices have to be fully implemented to achieve the benefits of XP. Boehm (2002) proposes that the migration towards usage of Agile methods is not so much a move from one end of the spectrum (plan-driven methods) to the other (agile methods), but rather can be thought of moving along a continuum. Hence, it is possible for organizations to adopt only those practices that best fit their needs and still achieve the benefits of agile methods.

eXtreme Programming is based on twelve key practices. Beck (2004) states that people can choose different practices to meet different situations. The continuous discussion on the usage and importance of each XP practice lead us to explore the following related research questions. What are the high priority XP practices that should always be followed? What practices are optional? Are there certain XP practices that increase project success? In this study, we surveyed practitioners in the field in order to answer these questions.

The remainder of this paper is organized as follows. We begin with a review of the eXtreme Programming literature. Next, we present our research method. Following this description, we present our research results and their implications. We conclude with a summary of the study's significant contributions.

eXtreme Programming Overview

One widely recognized Agile method is eXtreme Programming (XP). XP is a light weight software development methodology. XP is distinguished from other methodologies by short delivery cycles, incremental planning, collaboration among all the parties involved, emphasis on test first development, and flexibility to respond to changing business needs (Beck 2004). It is designed for small to medium skilled teams of up to 10 people that need to develop software quickly in an environment of vague or rapidly changing requirements (Beck 1999). Without a complex hierarchy of management structure, developers have more "control" of the project. It emphasizes productivity, flexibility, teamwork, minimal documentation and the limited use of technology outside of programming. XP promotes a discipline of software development based on four values; simplicity, communication, testing, courage and respect (Beck 2004).

The values of XP are implemented and enforced by the use of twelve principles listed in Table 1. These principles are core to the use of XP. Research suggests that following these principles leads to several advantages over traditional software engineering methods (Beck 2004). On the other hand, critics of XP state that the twelve principles are highly interdependent. Each principle can not stand up on its own because of its reliance on at least one other principle to support it. So, tailoring the XP principles is problematic (Stephans and Rosenberg 2004).

Table 1. Extreme programming twelve key principles (Beck 2000)

Principle	Principle Overview
Planning Game	Planning is continuous and progressive. Developers estimate the cost of the candidate features and customers select those features to be implemented based upon cost and business value.
Small Releases	Frequently releasing simple systems, and releasing new version on a very short cycle (1 to 3 weeks).
Simple Design	Keeping the system design as simple as possible and finding and removing extra complexity.
Test First Development	Writing and running unit tests as code is written to make sure the system is working properly. Frequent user acceptance tests to ensure the system is fulfilling user requirements.
Refactoring	A technique used to improve code without altering functionality. Focuses on simple, clean, non-repeating code that can be easily changed.
Pair Programming	Two programmers developing production code at the same time on one machine.
Collective Ownership	All developers responsible for all code, therefore changes to the code can be made by anyone at anytime when necessary.
Continuous Integration	Integrating new changes with current code as they are completed to detect system failures as soon as possible.
40-hour week	Developers keep a normal work schedule to remain productive and interested in the project.
On-site Customer	A customer sits with the development team full-time.
System Metaphor	Simple shared story of how the system works, to give both developers and customers common ground.
Coding Standards	Developers write all code in accordance with the standards agreed upon by the team to ensure that communication is made through code.

Many studies have been conducted to examine the effects of either a certain XP practice or a combination of various practices in different projects. We will present some of the key studies.

"Pair Programming is one of the most researched XP practice" (Abrahamsson and Koskela 2004). Both Williams (2003) and Parrish et al. (2004) conducted studies to examining the cost and productivity of pair programming. While Williams (2003) reported very positive numbers, Parrish et al. (2004) stated that "teams are more productive when their members work independently". However, they both acknowledged that pair programming improves code quality. Keefer (2005) listed numerous and severe flaws in Williams' experiments identified by several researchers (Bailey; Bromley and Higgs; Nawrocki et al. as cited in Keefer 2005). Keefer also suggested code reviews and inspections should be used instead of pair programming. People's opinions vary significantly on this practice.

Besides Pair Programming, another XP practice, On-site Customer, which is the only non-technical role on the XP team, is also "one of the most controversial topics in extreme programming methodology" (Abrahamsson and Koskela 2004). XP emphasizes having co-located customers to increase interaction since they fill the gap between developers and the business. They help clarify the projects' objectives and give constant feedback on the new functionalities of the code. In addition, the on-site customer gathers and prioritizes the requirements and makes decisions (Martin et al. 2004; Lippert et al. 2003). The rapid feedback from the customer improves the accuracy of the requirements and scenarios that will be implemented (Fruhling and Vreede 2006). However, the involvement of on-site customers may also bring distractions to the team. A controlled case study conducted by Abrahamsson and Koskela (2004) found that only 21% of on-site customers' efforts contributed to development, 42.8% to continuous planning and 19.9% to the acceptance test.

To our knowledge, there has been limited research on the use of System Metaphors practice. By using simple paragraphs and diagrams, System Metaphor is normally viewed as a communication tool to facilitate the shared understanding of how the system should perform (Robinson and Sharp 2003). Many people, however, are unfamiliar with this practice, and many others doubt the idea that the System Metaphor may take the place of "architecture design" (West and Solano 2004). Meanwhile, West and Solano (2004) also stated that the System Metaphor is a "ubiquitous and unavoidable" element of XP and more critical to project success than it was thought to be. However, Herbsleb et al. (2003) conducted two experimental studies indicating that System Metaphor is relatively useless.

The concept of Test First Development is increasingly popular (Nerur et al. 2005). In XP, the test cases are written before coding begins and testing occurs throughout the development process (Coram and Bohner 2005). Test First Development is intended to facilitate continuous integration, avoid scope creep, and bring confidence to project developers (Beck 2004; Nerur et al. 2005).

Other XP practices that have been examined are Simple Design and Planning Game. XP does not require the team to know everything at the beginning of the project (Beck 2004). By doing Simple Design and Planning Game, the team can start with a rough plan and modify the code in later iterations. Karlstrom (2002) found in his study that Planning Game, Collective Ownership and On-site Customers worked best together and contributed to greater project success.

While there are various XP research studies supporting the usage of XP and many organizations are implementing XP practices, still many people also express concerns with XP, such as lack of up-front design and general architecture (Rasmusson 2003; Cao et al. 2004), that mimics of hacker's culture (McBreen 2001) and lack of documentation and implementation difficulty for on-site customer (Paetsch et al. 2003). These concerns provide insights from other perspectives and suggest there is still more to learn on the implementation of XP practices.

Nevertheless, XP has been adopted by many software development teams. Some of them are within the original context that XP was designed (Fruhling and Vreede 2006). Meanwhile, several empirical studies reported that XP works well in large scale and legacy projects if it incorporates new elements and adapts to each situation appropriately (Lippert, et al. 2003; Boehm 2002; Cao et al. 2004).

In summary, eXtreme Programming has recently gained a significant amount of attention. While researches, controlled experiments and case studies have been conducted and discussed, less data is available regarding the actual usage and perceptions on the importance of each XP practice from practitioners in the field. In this study we researched, what are the indispensable XP practices to be implemented to achieve project success? In the following sections, we will discuss our research methodology, present our study, and try to find the answers for the questions mentioned.

Research Methodology

This study involved a cross-sectional anonymous survey of reported actual usage amount and perceptions of the importance of XP practices. Respondents were from local organizations that had already implemented XP into their projects. All items were simple phrases and measured using a seven-point Likert-type scale with anchors from "not used at all" or "not important at all" to "fully used" or "extremely important", accordingly.

Our research was conducted in companies located in the mid-west area of the U.S. in 2006. The surveys were either conducted on the premises of the participant's organization or sent out via US postal mail. It was important that the site surveyed had facilities available that were capable of supporting the administration and collection of anonymous survey responses. All participants were assured that their responses would be kept confidential; only research members had access to the data. We distributed 200 surveys, either in person or postage-paid mail, and 51 useful responses were collected. Out of

51 responses, 40 reported that they were using XP at their organizations. In the following section, we will perform our analysis based on the information provided by the 40 respondents who were using XP in their projects.

Results

Basic Demographics

Demographic data gathered in this study consisted of roles in the company, team size, work experience, and industry type. Besides that, we were also interested in the way that XP practitioners communicate with each other. The subjects in this research have the following characteristics. They were professionals who were working for IT firms or departments in the mid-west area. They were using or at least have used some of the XP practices at their work. Over 60% of respondents indicated that they have over ten years IT work experience. Their organization primarily operated in the financial industry (banking, credit card, etc.), and 65% of them worked in teams of between 10 and 15 members. Almost 60% of the respondents indicated their positions as developers or programmers. Responses also came from other positions, i.e., manager, Q.A. business analyst. However, each of these categories made up around 10% of the respondents population. Demographic features of the surveyed population are shown in Table 2.

Table 2. Demographic Information

Work Experience	**Responses**
Under one year	1
Between one and three years	2
Between three and five years	3
Between five and ten years	9
Over ten years	25
Industry Type	
Financial	37
Insurance	1
Manufacturing	2
Team Size	
1 – 5	3
5 – 10	7
10 – 15	26
15 – 25	4
Roles	
Manager	3
Architect	3
Business Analyst	4
Developer	23

Other	7

In terms of communication, most respondents indicated that they have morning standup meetings and that they prefer face-to-face communications. It is worth noting that a certain number of them replied that they were using a Wiki as a means of communication, which is a piece of server software that allows users to freely create and edit web page content using any web browser. In summary, their work environment is characterized as being collaborative, access to fast and convenient tools (i.e., e-mail, wiki), and direct but informal interactions (i.e., morning stand-up meetings, face to face) that facilitate the communication among members on the XP team.

Practice Usage versus Practice Importance

Actual Usage of XP practices

The first survey question we asked was "*Please indicate the usage amount of each of the following practices in your system development project. (1 = not used at all; 7 = fully used)*" The following table lists the mean value of usage amount of each XP practice indicated by practitioners from low to high. The corresponding Standard Deviation (σ) value of each practice is also listed. The mean of the responses indicates the central tendency of the population on a certain issue. Standard Deviation (σ) value describes how and how much the responses differ. The higher the σ value, the greater the responses vary.

From Table 3, we can find the usage amounts of the following four practices: Collective Ownership, Continuous Integration, Pair Programming, and Planning Game are the highest. The σ values for Collective Ownership (σ =0.89), Continuous Integration (σ =0.899), and Sustainable Pace (σ =0.871) are very low, which indicate a small variance for the usage amount of the three practices. Compared with other practices, the mean for the usage amount of System Metaphor is the lowest and its σ value (1.894) is also the highest. Therefore, there is not as much consensus among the respondents.

Table 3. Mean and σ of each XP Practice in Terms of Actual Usage Amount

XP Practice Usage	Mean	Standard Deviation (σ)
System Metaphor	4.13	1.894
On-site Customer	4.75	1.691
Refactoring	4.9	1.429
Coding Standards	4.9	1.501
Simple Design	5.05	1.239
Test First	5.35	1.406
Small Release	5.63	1.03

Sustainable Pace	5.9	0.871
Planning Game	6	1.24
Pair Programming	6.08	1.575
Continuous Integration	6.25	0.899
Collective Ownership	6.35	0.893

Importance of XP practices

Besides the actual usage amount of each XP practice in the field, we were also interested in practitioners' perceptions on the importance of each practice. The question we asked was "*Please indicate how important each attribute is to the success of an I.S. project*"; 7 indicates extremely important, and 1 indicates extremely unimportant. Table 4 lists practitioners' perception of importance of each practice to achieve project success; the σ values are also listed accordingly.

Table 4. . Mean and σ of each XP Practice in Terms of Perceived Importance

XP Practice Importance	Mean	Standard Deviation (σ)
System Metaphor	4.77	2.019
Coding Standards	5.33	1.221
Pair Programming	5.65	1.642
Simple Design	5.98	1.121
Small Release	6.05	1.108
Refactoring	6.08	1.207
Sustainable Pace	6.08	1.047
On-site Customer	6.1	1.277
Collective Ownership	6.13	0.883
Planning Game	6.25	1.032
Test First	6.45	0.904
Continuous Integration	6.51	0.79

The importance of Planning Game (6.25) and Continuous Integration (6.51) are rated very high in this circumstance. Surprisingly, we found that Test First Development (6.45) and On-Site Customer (6.10) were also rated very high by practitioners. This indicates that practitioners acknowledged their contributions to a project's success. System Metaphor is still rated as the lowest with the highest σ value in this situation, even though the mean is 4.77. It was unexpected that the importance of uniform Coding Standards (5.33) is rated the second lowest.

Comparison between usage amount and importance

We compared the means between actual usage amount and perception of importance of each XP practice respectively, based on the convenience responses (responses that indicated XP was being used in their current project) in Figure 1. The means of usage amount of each XP practice are sorted from low to high, and the means of perceived importance vary along this line. Generally speaking, the ratings of importance are higher than their actual usage amount

respectively, except Pair Programming and Collective Ownership. However, they are among the highest ratings, both in terms of perception of importance and usage amount.

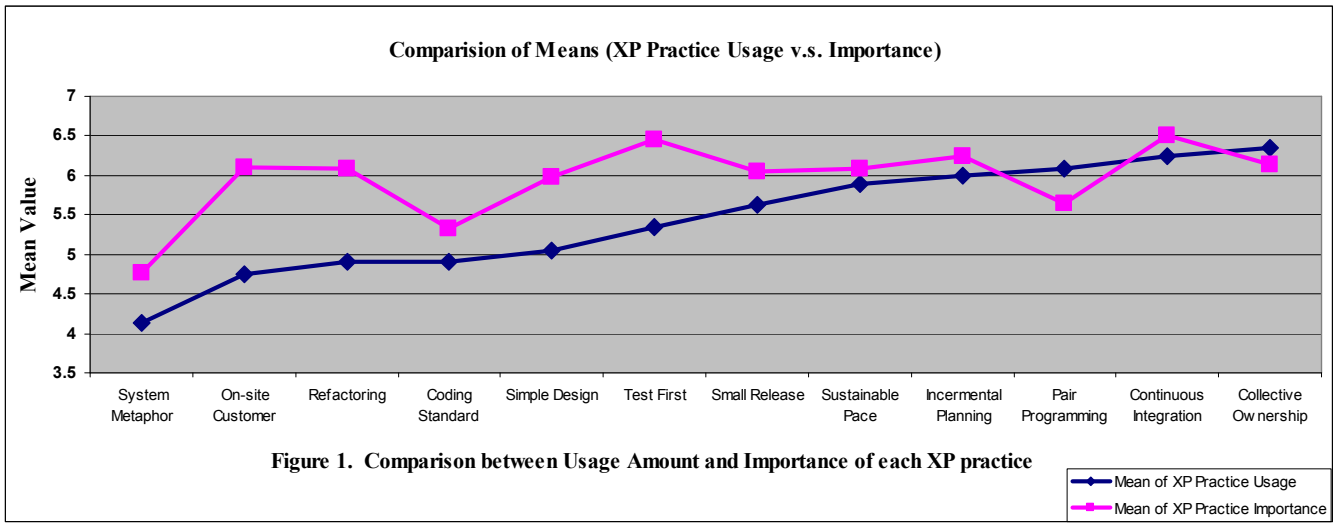

Figure 1. Comparison between Usage Amount and Importance of each XP practice

The Paired Sample t Test is used to determine whether there is a significant difference between two variables. In addition, the results also answer if there are correlations between two variables. In this research, the paired sample t-test was computed on each XP practice between actual usage amount and perceived importance to project success. Pair 4, for example, compared the variable "Test First Usage" whose mean is 5.35 with the variable "Test First Importance" whose mean is 6.45 (see Table 3 and 4). Table 5 contains the list of six pairs that showed significant correlations with lowest sig value. The correlation is more marked on Planning Game (Pair 1: 0.701) and On-site Customer (Pair 10: 0689), which may indicate that the perception on these two practices has strong positive correlation on their actual usage. All of the pairs in Table 5 demonstrate a relationship between the actual usage amount and the perceived importance by the user.

Table 5. Significant Paired Sample Correlations

		N	Correlation	Sig.
Pair 1	Planning Game Usage & Planning Game Importance	40	**0.701**	0.000
Pair 4	Test First Usage & Test First Importance	40	0.498	0.001
Pair 5	Refactoring Usage & Refactoring Usage Importance	40	0.555	0.000
Pair 6	Pair Programming Usage & Pair Programming Importance	40	0.576	0.000

Pair 10	On-Site Customer Usage & On-Site Customer Importance	40	**0.689**	0.000
Pair 12	Metaphor Usage & Metaphor Importance	39	0.579	0.000

Note: Only Sig. < 0.002 are shown above.

As shown in Table 6, the results show a significant difference (sig < 0.05) between actual usage and perceived importance on the following five practices: Simple Design, Test First Development, Refactoring, On-site Customer, and System Metaphor. For all of the five practices, the means of the perceived importance are much higher than the means of actual usage amount respectively. From here we may infer that the following five practices should be used more, and more attention should be paid to them.

Table 6. Paired Sample t test

	Paired Differences					t	df	Sig. (2-tailed)
	Mean	Std. Deviation	Std. Error Mean	95% Confidence Interval of the Difference Lower	Upper			
Pair 3 Simple Design Usage – Simple Design Importance	-0.925	1.591	0.252	-1.434	-0.416	-3.676	39	0.001
Pair 4 Test First Usage – Test First Importance	-1.1	1.236	0.195	-1.495	-0.705	-5.628	39	0.000
Pair 5 Refactoring Usage – Refatoring Importance	-1.175	1.259	0.199	-1.578	-.772	-5.905	39	.000
Pair 10 On-site Customer Usage – On-site Customer Importance	-1.35	1.231	0.195	-1.744	-0.956	-6.936	39	0.000
Pair 12 Metaphor Usage – Metaphor Importance	-0.641	1.799	0.288	-1.224	-0.058	-2.225	38	0.032

Note: Only Sig. (2-tailed) < 0.05 are shown above.

Discussion

Based on the results presented in the last section, we selected the highest and lowest values (means for actual usage amount, means for perceived importance, and their standard deviations respectively) and present only them in Table 7. Thus, making it is easier to analyze the noticeable differences between practices. Next, we compared the findings of our research with other studies mentioned in the literature review.

Table 7. Values that Stand out

Practice	Usage		Standard Deviation of Usage		Importance		Standard Deviation of Importance	
	High	**Low**	**High**	**Low**	**High**	**Low**	**High**	**Low**
System Metaphor		4.13	1.89			4.77	2.02	
On-site Customer		4.75	1.69		6.10		1.28	
Refactoring		4.90						
Coding Standards		4.90				5.33		
Simple Design								
Test First					6.45			0.90
Small Release	5.63							
Sustainable Pace	5.90			0.87	6.08			
Planning Game	6.0				6.25			
Pair Programming	6.08		1.575			5.65	1.64	
Continuous Integration	6.25			0.9	6.51			0.79
Collective Ownership	6.35			0.89	6.13			0.88

Based on the responses from surveyed practitioners, the actual usage amount of Pair Programming (6.08) was much higher than practitioners' perceptions on its importance (5.65). The Standard Deviation (σ) values of Pair Programming were high as well, both in terms of actual usage amount (1.58) and importance to success (1.64), which indicate the discrepancies among practitioners. This practice was used quite often in the field but its contribution for project success was not agreed upon by practitioners. Though the implied actual usage amount of Collective Ownership (6.35) is also higher than its perceived importance (6.13), the difference is not obvious. Small σ values of Collective Ownership, together with high mean values, both in terms of usage amount and perceived importance, indicate that Collective Ownership is very essential for XP project from the perspective of practitioners.

Generally speaking, the usage of Continuous Integration (6.25), Sustainable Pace (5.9), and Collective Ownership (6.35) is quite high in the field. The importance of Continuous Integration and Collective Ownership for project success is agreed upon among practitioners. However, the perception on the importance of Sustainable Pace to project success varies among practitioners, but having a normal schedule (normal pace/40hour per week) is still thought to be important and has been followed by most companies.

The mean of actual usage amount of On-site Customer (4.75) was low relative to its mean of perceived importance (6.10) for project success. Research subjects' perception, however, vary greatly on this practice (σ =1.28), therefore the implied usage also varies (σ =1.69), Practitioners agreed on the importance of On-site Customers for project success and may expect more customer involvement; however, there might be many difficulties that impede the full implementation of this practice. The importance of On-site Customer was emphasized by practitioners. Refactoring is in a similar situation as On-

site Customer. Practitioners may find this technique hard to grasp and/or implement, but its importance for project success is recognized.

The current usage amount of Test First Development (5.35) did not stand out, but the importance of Test First Development to project success was rated very high (6.45) by surveyed practitioners, and the responses had a small variance (σ =0.9). As mentioned in the literature review section, Test First Development can facilitate other XP practices and bring confidence to the team, which is important during the software development process. Data collected in this research shows that most XP projects paid attention to testing, and more efforts are still needed.

The data collected on System Metaphor is as controversial as reflected in the literature review. Implied by XP practitioners in our research, System Metaphor has very low usage (4.13) in their projects generally. Some practitioners think it is very important to implement this practice while others indicate that it is almost useless. The contradiction leads us to wonder if people misunderstand this concept, or if it is not found to be useful.

Conclusion

The study of XP methodology is still far from over. However, we need to expand our focus by looking at the actual usage and perceptions from practitioners. This preliminary study has demonstrated that there is a significant relationship between them. That opens the areas for future research. One potential area to examine is the variation of practices across different industries. Another more fundamental inquiry is to explore why there are significant differences between actual usage and perceptions of importance on some practices.

Auer and Miller (2002, p.68) reported the essential practices when you initiate an XP project as follows: planning and estimating, small releases (and iterations), testing first (unit testing), pair programming, refactoring and continuous integration. It is not that the other practices are unimportant, but they can wait until the practices listed above are in place.

In some cases our research agrees with Auer and Miller's (2002) essential practices and in other cases is does not. Collective Ownership was almost fully used and it was rated as very important in our research where as it did not appear in the essential practices list. In this study, both Planning Game and Continuous Integration stand out for their significant usage amount and perceived importance for project success, which does overlap with the list above. Practitioners responses imply that more attention should be made to Test First Development as indicated in the results that its rated importance is much higher than current usage amount.

Practitioners' opinion on System Metaphor was in accordance with past studies. This practice was used the least and its importance was not acknowledged in our study. The perceptions on this practice have large variances. The

usefulness and importance of On-site Customer was emphasized by practitioners, though it seems there are some difficulties in actual implementation. Further investigation is warranted.

Contribution

To the best of our knowledge, this is the first study to empirically articulate the actual usage amount, and perceptions of importance of the twelve XP practices. Survey questions effectively collected local practitioners' responses. Within the two categories (actual usage amount and perceptions of importance), we ranked twelve practices accordingly. When comparing across these two categories, the significant correlations and differences of practices were also determined. We analyzed the important implications for both practitioners and researchers.

Due to the limited time and sample size, this cross sectional research may only contribute to mid-western organizations in the financial industry. Further research is needed to confirm whether the implications in this study can be applied to businesses in different industries, or whether different industries are unique and have different implementation of XP practices. However, this study provides a solid foundation for future research on XP implementation and could potentially be used by all practitioners.

References

Abrahamsson, P., and Koskela, J. "Extreme Programming, A Survey of Empirical Data from a Controlled Case Study," ACM-IEEE International Symposium on Empirical Software Engineering (ISESE 2004*)*, Redondo Beach, 2004, CA, USA.

Auer K., &Miller, R. Extreme Programming Applied. Boston, MA: Addison-Wesley, 2002.

Bailey, R. "A Critical look at some Pair Programming Research." Hacknot Web Site, 6 April, 2004. (http://www.hacknot.info/hacknot/action/showEntry?eid=50).

Beck, K. "Embracing Change with Extreme Programming," Computer (32:10), 1999, pp.70-77.

Beck, K. Extreme Programming Explained: Embrace Change, MA Boston: Addison-Wesley, 2000.

Beck, K. and Andres, C. Extreme Programming Explained: Embrace Change (second ed.), MA Boston: Addison-Wesley, 2004.

Boehm, B. "Get Ready for Agile Methods, With Care," Computer (35:1), 2002, pp.64-69.

Bromley, Higgs, How not to Perform a Pair Programming Experiment, *Student Position Paper,* University of Sheffield.

Cao, L., Mohan, K., Xu, P., and Ramesh, B. "How Extreme does Extreme Programming Have to be? Adapting XP Practices to Large-scale Projects," Proceedings of the 37th. Hawaii International Conference on System Science, 2004.

Coram, M., and Bohner, S. "The Impact of Agile Methods on Software Project Management," Proceedings of the 12th IEEE International Conference and Workshops on the Engineering of Computer-Based Systems (ECBS'05), 2005.

Fruhling, A., and De Vreede, G-J. "Field Experiences with eXtreme Programming: Developing an Emergency response System," Journal of Management Information Systems (22:4), 2006, pp.39-68.

Herbsleb, J. Root, D., and Tomayko, J. The eXtreme Programming (XP) Metaphor and Software Architecture, CMU-CS-03-167, 2003, Carnegie Mellon, Pittsburgh, PA, USA

Karlstrom, D. "Introducing eXtreme Programming - An experience report," Third International Conference on eXtreme Programming and Agile Processes in Software Engineering (XP'2002), Sardinia, Italy, 2002, pp 24-29.

Keefer, G. "Pair Programming: An Alternative to Reviews and Inspections?" Cutter IT Journal (18:1), 2005, pp.14-19.

Lippert, M., Becker-Pechau, P., Breitling, H., Koch, J., Kornstadt, A., Roock, S., Schmolitzky, A., Wolf, H., and Ziillighoven, H. "Developing Complex Project Using XP with Extensions," Computer (36:6), 2003, pp.67-73.

Martin, A., Biddle, R., and Noble, J. "The XP customer Role in Practice: Three Studies," Proceedings of the Agile Development Conference, 2004, pp.42-54.

McBreen, P. Questioning extreme programming. New York: Addison-Wesley, 2001.

Nawroki, J, and Wojciechowski, A. "Experimental Evaluation of Pair Programming," Paper presented to the 12th European Software Control and Metrics Conference, 2001.

Nerur, S., Mahapatra, R., and Mangalaraj, G. "Challenges of Migrating to Agile Methodologies," Communication of the ACM (48:5), 2005, pp. 73-78.

Paetsch, F., Eberlein, A., & Maurer, F. Requirements Engineering and Agile Software Development. Proceedings of the Twelfth IEEE International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises (WETICE'03), 2003.

Parrish, A., Smith, R., Hale, D., & Hale, J. "A Field Study of Developer Pairs: Productivity Impacts and Implications," IEEE Software (21:5), 2004, pp.76-79.

Rasmusson, J. Introducing XP into Greenfield Projects: Lessons Learned. IEEE Software (20:3), 2003, pp 21-28.

Robinson, H., & Sharp, H. XP Culture: Why the twelve practices both are and are not the most significant thing. Proceedings of the Agile Development Conference (ADC'03), 2003.

Stephens,M. and Rosenberg, D. "The irony of extreme programming," Dr. Dobbs Journal, May, 2004, pp 44-47.

West, D., and Solano, M. "Metaphors be with you!," Proceedings of the Agile Development Conference (ADC'05), 2005.

Williams L., McDowell, C., Nagappan, N., Fernald, J., and Werner, L. "Building Pair Programming Knowledge through a Family of Experiments," Proceedings of the International Symposium on Empirical Software Engineering, 2003, pp. 143 – 152.