

X THE IMPACT OF METHODS AND TECHNIQUES ON OUTCOMES FROM AGILE SOFTWARE DEVELOPMENT PROJECTS

David Parsons
Hokyoung Ryu
Ramesh Lal
*Massey University, Albany
Auckland, New Zealand*

Abstract

Agile software development methods have become increasingly popular since the late 1990s, and may offer improved outcomes for software development projects when compared to more traditional approaches. However there has previously been little published empirical evidence to either prove or disprove this assertion. A survey carried out in March 2006 gathered responses from a large number of software development professionals who were using many different methods, both traditional and agile. A statistical analysis of this data reveals that agile methods do indeed improve outcomes from software development projects in terms of quality, satisfaction, and productivity, without a significant increase in cost. However, adoption of methods appears to involve a high degree of adaptivity, with many methods being used in combination and sets of techniques being adopted on an ad hoc basis. In this context, our analysis suggests that choosing specific combinations of methods can be particularly beneficial. However, we also find that successful adoption of an agile method is to some extent dependent on rigorous integration of certain core techniques.

Keywords

Agile method, technique, software development

Parsons, D., Ryu, H., and Lal, R., 2007, in IFIP International Federation for Information Processing, Volume XXX, Organizational Dynamics of Technology-Based Innovation: Diversifying the Research Agenda, eds. McMaster, T., Wastell, D., Ferneley, E., and DeGross, J. (Boston: Springer), pp. xx-xx.

1 INTRODUCTION

This paper provides some statistical analyses of a data set originally gathered using an online survey to determine the level of adoption of agile approaches by software development organizations. The survey, carried out in March 2006 by Scott Ambler, used mailing lists from both *Dr. Dobbs's Journal* and *Software Development* magazine. A summary article was subsequently published on-line in September 2006 (Ambler, 2006c), and the raw data was made available for public access (Ambler 2006a). In this paper we view the original data from a number of new perspectives to explore some important questions about the effects of some key variables on the outcomes of software development projects. We begin by looking at previous studies that relate to the adoption and adaptation of agile methods and techniques. We then introduce the data set and methodology that we have used in this study. This is followed by an analysis of the data, from which we draw some conclusions and propose some further work.

2 AGILE METHODS AND TECHNIQUES

The mid-1990s saw the emergence of a set of informal analysis and design approaches known as *agile methods* (Highsmith 2002). While proponents of agile methods claim software development improvements, there is little empirical evidence to back this claim even though “agilists” emphasize that benefits would be experienced if these methods and practices were used (Anderson 2004).

2.1 Adopting and Adapting Agile Methods

Although agile methods tend to be quite prescriptive about the practices that they do or do not include, there is some information suggesting that methods and techniques are being adopted in a somewhat piecemeal manner (Aveling 2004; El Emam 2003; Hussman 2006). This is not inconsistent with the *intent* of these methods, since it is recognized that each software development project differs in its scale, scope, and technical challenges. Therefore agile methods encourage the chosen approach to be adapted to counter the various development conditions that apply to a particular development project (Keenan 2004; Mišić 2006). Integrating any new practice or software development process requires method tailoring to integrate it with existing processes and to match the organizational environment (Lindvall et al. 2004; Sfetsos et al. 2006). Sometimes it is necessary to stage the introduction of certain techniques because of dependencies between them (Beck and Andres 2005), while some authors propose extensions to certain methods to compensate for what they regard as their limitations. For example Stephens and Rosenberg (2003, p. 380) describe eXtreme Programming as an “anorexic process without effective contingency plans” and suggest a significant *refactoring* of the method. Others suggest that combining multiple agile methods may be more effective than using one method alone (Beedle 2006; Mar and Schwaber 2002). Choosing an appropriate method may also be problematical in practice. Datta (2006) proposes a framework known as the Agility Measurement Index (AMI) which can be

used to determine the appropriateness of an agile method for a particular software development project, while Domingues et al. (2006) suggest a *suitability filter* for selecting specifically from agile methods. However, most software development teams do not have the depth of knowledge and skills to pick and choose different methods for different projects. The normal practice for software development is to adopt the most convenient or familiar method and then evaluate and improve it as it is being used to develop applications and systems. The implication of these issues is that methods may be adopted in ways that lead to extensive tailoring. This tailoring is likely to mean variations in the numbers and types of techniques adopted regardless of the label used to describe the umbrella agile method, or methods.

2.2 Choosing and Using Agile Techniques

Previous research has indicated that the selection of agile techniques within a method may be influenced not only by their perceived benefits but also by a range of problems and difficulties that certain techniques bring with them. A study by Sfetsos et al. (2006) identified pair programming and test-driven development to be the most significant success factors in outcomes from agile development, but noted that companies found problems adopting some other techniques such as common code ownership, on-site customer, 40-hour week, and metaphor. In contrast, Grossman et al. (2003) suggested that test-driven development was the most difficult and risky technique to adopt, as did a set of experiments at three different locations by George and Williams (2003). While many studies have highlighted the benefits of test-driven development, including defect reduction (Williams et al. 2003), a better testing process (Dustin et al. 1999) and higher quality (Bhat and Nagappan 2006), there are also some suggested drawbacks, such as a slower overall process (Bhat and Nagappan 2006; Canfora et al. 2006). Indeed some have suggested that this technique does not even, in fact, improve code quality (Muller and Hagner 2002). Further studies suggest that test driven development is not the only problematic technique. Others include simple design, pair programming, customer tests, and collective code ownership (George and Williams 2003; Mišić 2006). Paige et al. (2005) discuss some negative issues relating to increments, pair programming, and customer feedback in the context of building high-integrity systems (HIS). These studies show that there are many social, organizational, and technical factors that may influence why an agile technique may or may not be used by a particular software development team. This may explain why agile methods are not always adopted in full, but rather that certain techniques are adopted and methods are adapted.

Clearly, adopting an agile method in practice is not simply a case of taking a single method off the shelf and adopting its practices. Rather, it involves a process of selection and adaptivity. Given that practitioners are adopting various combinations of methods and techniques, the question that we try to address in this paper is which methods and techniques appear to provide the best outcomes. In the ongoing debate about the wisdom or otherwise of embracing agile methods (Boehm 2002; Nerur et al. 2005), empirical evidence such as the survey data used in this paper can make a valuable contribution to our understanding and assist software developers in building an appropriate methodology from the various agile methods and techniques on offer.

3 THE AGILE ADOPTION SURVEY DATA

The data set used in this paper was made available by Scott Ambler (2006a) and is based on an on-line survey. The Ambler survey repeated, with some changes, a similar survey carried out by Shine Technologies (2003). This original survey had only 131 respondents but Ambler's survey had 4,235 respondents.

Perhaps the most important aspect of both questionnaires is the four questions relating to the outcomes of software development projects, namely; productivity, system quality, cost, and stakeholder satisfaction. Ambler (2006c) endeavored to improve the original Shine survey in a number of ways. One important difference is that some Likert scale responses also included "I don't know" as a response for the four questions that related to outcomes, making it possible to discount these responses from our analyses. What is most interesting about the Ambler survey is that it introduced questions about the agile techniques that were being adopted, making it possible to do some analysis of which techniques were actually being used within the various methods. It also made it possible to see if the use of certain practices could be correlated with certain outcomes. In this paper, to explore the relationship between outcomes and agile methods and techniques, we focus on the outcomes as dependent variables, with the use of methods and techniques as independent variables.

In his original article, Ambler (2006c) drew some preliminary conclusions from the data. For instance, there was a correlation between knowledge of agile development and outcomes, so that the respondents who were more knowledgeable about agile approaches claimed to have better quality, productivity and satisfaction than those who were not. He also concluded that organization size was not a statistically significant factor in the levels of outcome attained from agile approaches. The most significant result claimed was that adoption of agile methods increases quality, productivity, and satisfaction, and that "adoption of agile processes has clearly been a resounding success" (Ambler 2006c, p. 3).

We do not intend to replicate all of these analyses in this paper, but prefer to explore additional features of the data. Ambler (2006c) did not attempt to provide a deeper analysis of any of the key variables that may have effects on the adoption of agile methods. This gave us an opportunity to mine the data set for some further insights into the effects of adopting agile methods and techniques, which is the main concern of this paper.

Looking at the data, we noted that many organizations are using more than one agile method. In fact nearly 16 percent of the respondents claimed to be using multiple methods, in some cases as many as seven (Table 1). This raised an obvious question about whether there is any benefit in adopting multiple methods for agile development.

We also noted that techniques and methods did not seem to be consistently used as one might expect with, in many cases, a lack of correlation between the stated use of a method and actual use of techniques that would normally be associated with that method. In other words, there were no consistent patterns between the stated methodology and the techniques actually being used. The 12 techniques included in the survey, and the number of respondents reporting their use, is shown in Table 2.

The lack of correlation between specific techniques and methods in practice led us to question: What is the underlying relationship between a methodology and a set of techniques?

Table 1. The Numbers of Agile Methods Reported as Being Used by the Survey Respondents

Number of Methods Used	Number of Respondents	Percentage of Respondents
No agile methods	2541	59.99%
One agile method	1019	24.06%
Two agile methods	500	11.80%
Three or more methods	175	4.15%
Total	4235	100%

Table 2. The Numbers of Agile Techniques Reported as Being Used by the Survey Respondents

Techniques	Number of Respondents	Percentage of Respondents
Active stakeholder participation	938	22.15%
Agile model driven development	260	6.14%
Code refactoring	1467	34.64%
Code regression testing	1383	32.66%
Colocation	447	10.55%
Common coding guidelines	1595	37.66%
Continuous integration	1113	26.28%
Database refactoring	416	9.82%
Database regression testing	407	9.61%
Pair programming	587	13.86%
Single sourcing	241	5.69%
Test driven design	959	22.64%

Given the data set provided, we were able to propose some initial, broad research questions that might be answered by this data. These research questions were

- Does the use of agile methods have a positive effect on outcomes (i.e., cost, productivity, quality, and satisfaction)?
- What are the most effective agile methods?
- What are the most effective agile techniques?

The following sections detail how these questions were addressed, using statistical analyses, in order to try to identify the most important success factors in agile development.

4 DATA ANALYSIS

We performed quantitative analyses of parts of the data set. To do this, the original data was re-coded and analyzed using SPSS (version 13.0). It was originally planned to use an analysis of variance, but Levene’s test for heterogeneity of variance was found to be significant in most of the following analyses, suggesting that the data were not in fact suitable for analysis of variance. It would have been possible to transform the data but this would have made it difficult to interpret. For these reasons, the simple solution of using nonparametric analyses was adopted.

Several notes about the data set itself are needed here. First, the “don’t know” responses from the original data have been regarded as “missing values” in the analysis. As a consequence, there are variations in the sample sizes in each analysis. Second, there were very small sample sizes for some responses, for example the dynamic systems development method (DSDM) was being used by only a handful of respondents. Finally, as mentioned above, quite a few respondents were using more than one agile method and various combinations of techniques (see Tables 1 and 2), so it is not straightforward to separate out the effects of a particular method or technique. For this reason, we chose to analyze both methods and techniques separately, and then if a particular agile method appeared to be the most effective, we planned to investigate the relationship between the method and the appropriate techniques in depth.

4.1 Question 1: Does the Use of Agile Methods Have a Positive Effect on Outcomes?

The first question we addressed in our analysis was whether the adoption of agile methods for software development might lead to better outcomes (i.e., in cost, productivity, quality, and satisfaction), as much of the literature on agile development has claimed. To do this we first explored the four outcomes, contrasting the non-agile method user group with the agile user group. The results of this analysis are shown in Table 3.

The results seem to indicate a positive response for the agile methods the respondents have employed. That is, all the three performance-related outcomes (i.e., productivity, quality, and satisfaction) indicate the benefits of agile methods, while there seems to be no great difference with regard to cost.

Table 3. The Effect of Using an Agile Methodology[†]

	Cost	Productivity	Quality	Satisfaction
No agile methods	3.01 (0.62)	3.40 (0.70)	3.55 (0.74)	3.40 (0.70)
Agile methods	3.05 (0.89)	3.88 (0.79)	4.02 (0.77)	3.95 (0.79)
Mann-Whitney U test	n.s.	p < .01	p < .01	p < .01

[†]Mean (s.d.) (min: 1 – much lower, max: 5 – much higher)

Table 4. Outcomes Dependent on the Number of Agile Methods Used[†]

	Cost	Productivity	Quality	Satisfaction
One agile method	3.03 (0.85)	3.83 (0.78)	3.98 (0.76)	3.89 (0.75)
Two agile methods	3.06 (0.92)	3.92 (0.77)	4.06 (0.76)	3.98 (0.82)
More than two agile methods	3.12 (1.03)	4.01 (0.86)	4.14 (0.80)	4.14 (0.84)

[†]Mean (s.d.)

This observation has been confirmed by the Mann-Whitney U test as shown in the bottom row of Table 3. Only the difference in the cost factor was not found to be statistically significant, supporting the interpretation given above.

Following on from this, a subsequent analysis was performed to identify what agile methods seem to be the most effective. To do this, we have to look at the data set with care. In many cases, the agile method user group reported that they used more than one agile method, so some developers are combining several agile development methods, either in different projects or within the same project. Therefore, to explore which agile methods are the most effective, we first investigated if any combinative use of agile methods (or, at least, employing more than one agile method) can have an effect on the four outcomes. The number of methods the respondents were using varied in the original data set, ranging from one to seven. However, to allow the sample size to be meaningful for statistical analysis, three classifications were considered: using one agile method, using two agile methods, and using more than two agile methods. Our results are summarized in Table 4.

In terms of the three performance-related outcomes (i.e., productivity, quality, and satisfaction), as shown in Table 4, it appears to be better to use more than two agile methods (mean 4.01 for productivity, 4.14 for quality, and 4.14 for satisfaction). However, further pair wise Mann-Whitney tests (at the level of $p \leq .05$) revealed that while using two agile methods rated higher than using only one method, there appeared to be no significant further advantage in increasing the number of methods used beyond two.

4.2 Question 2: What Are the Most Effective Agile Methods?

Following on the result described above (i.e., that combining two agile methods might be beneficial in agile software development), we continued to analyze the results for adopting combinations of two agile methods. It was intended to see if any specific pair of methods could deliver better outcomes than the others. There were 24 different method pairs identified in the original data set, but many of these pairs were being adopted by a very small number of respondents, which took them beyond the scope of our data analysis. Figure 1 shows the 11 most commonly used pairs of agile methods, where more than 10 respondents reported using these pairs. However, only the most popular 6 pairs (with over 20 respondents) were considered in our analysis.

For this data, we applied the same nonparametric analysis (i.e., pairwise Mann-Whitney U tests) as we did with the previous set. The results of this analysis are shown in Table 5. In terms of both quality and productivity, there was a significant difference

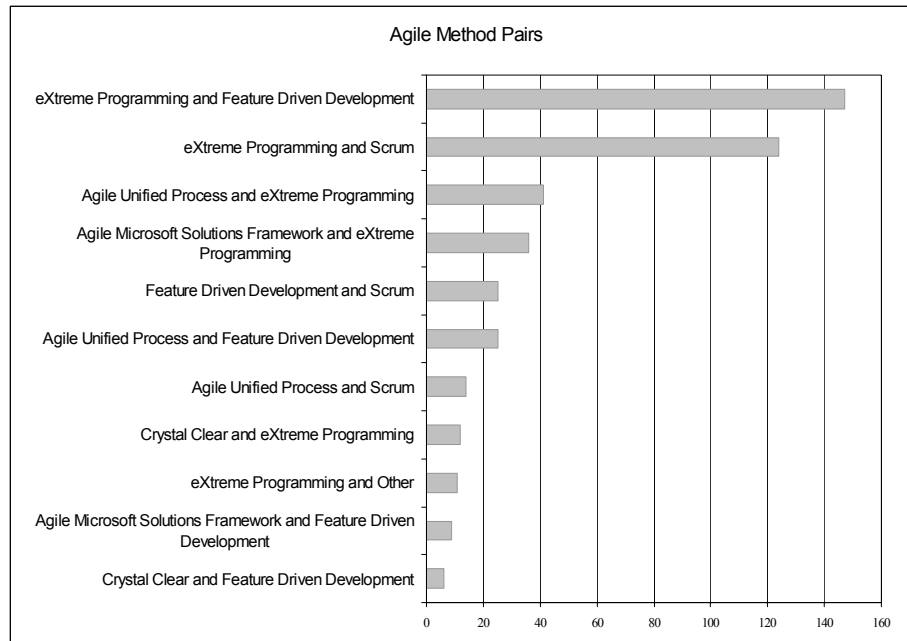


Figure 1. The Most Commonly Used Pairs of Agile Methods

Table 5. Outcomes for the Six Most Commonly Used Pairs of Agile Methods[†]

	Cost	Productivity	Quality	Satisfaction
XP/FDD	3.02 (0.89)	3.87 (0.70)	4.10 (0.75)	4.03 (0.77)
XP/SCRUM	2.91 (1.04)	4.10 (0.80)	4.30 (0.68)	4.05 (0.86)
XP/AGILE UP	3.26 (0.75)	3.87 (0.80)	3.82 (0.94)	3.97 (0.74)
XP/AGILE MSF	3.17 (0.95)	3.56 (0.82)	3.78 (0.87)	3.82 (0.73)
FDD/SCRUM	3.15 (0.88)	4.00 (0.69)	3.91 (0.53)	4.29 (0.78)
FDD/AGILE UP	3.00 (0.66)	4.04 (0.55)	4.00 (0.78)	3.79 (0.72)

[†]Mean (s.d.)

between the eXtreme Programming/Scrum combination and all the other pairs of methods. However there was no significant difference in either cost or satisfaction. This clearly tells us that the eXtreme Programming/Scrum combination is a good pairing of methods to adopt.

This result can be seen to make some sense in that eXtreme Programming (XP) is very much oriented towards technology based practices and programmer activity. In contrast, Scrum is more focused on agile project management aspects (Abrahamsson et al. 2002). In addition, Scrum is explicitly intended as a wrapper around other engineering approaches. Therefore XP and Scrum can be seen to be complementary from a practical point of view, supporting the claims made by Mar and Schwaber (2002).

Table 6. Agile Techniques with Significant Benefits for Software Development Outcomes[†]

	Productivity	Quality	Satisfaction
Active stakeholder participation	3.92 (0.76)	4.08 (0.73)	4.07 (0.76)
Agile model driven development	3.93 (0.82)	4.03 (0.82)	4.03 (0.79)
Code refactoring	3.91 (0.74)	4.09 (0.71)	3.92 (0.78)
Code regression testing	3.84 (0.75)	4.05 (0.71)	3.89 (0.77)
Colocation	3.99 (0.79)	4.08 (0.76)	4.00 (0.83)
Common coding guidelines	3.79 (0.76)	3.98 (0.73)	3.86 (0.77)
Continuous integration	3.97 (0.74)	4.11 (0.73)	3.99 (0.77)
Database refactoring	3.88 (0.78)	4.05 (0.74)	3.96 (0.83)
Database regression testing	3.78 (0.76)	3.98 (0.74)	3.86 (0.80)
Pair programming	3.97 (0.77)	4.15 (0.75)	3.99 (0.78)
Single sourcing information	3.93 (0.80)	4.00 (0.80)	4.00 (0.81)
Test driven design	3.95 (0.76)	4.18 (0.70)	4.01 (0.77)

[†]Mean (s.d.)

4.3 Question 3: What Are the Most Effective Agile Techniques?

Having undertaken some analysis of the effects of method choice on outcomes, we turned our attention to individual agile techniques. Since there are 12 different agile techniques covered in the original data, it was interesting to see if any of these provided greater benefits than others. The results of our analysis are shown in Table 6. The cost factor was eliminated in this analysis, because our main concern was to see what positive benefits a particular agile technique could bring.

To analyze this data, we first applied the same statistical approach that was used in the previous analysis. However, the ratings were uniformly high. Because of this overall ceiling effect, the small differences between means are not statistically evaluated. Nonetheless, several of the mean ratings given in Table 6 show a particularly interesting aspect. The techniques of colocation (3.99 in productivity, 4.08 in quality, and 4.00 in satisfaction, respectively) and pair programming (3.97 in productivity, 4.15 in quality, and 3.99 in satisfaction, respectively) appear to bring higher benefits for all three outcomes. Without further research, we cannot say for sure why these techniques appear to provide higher returns than some of the others. However, we can see support in the literature for the economic benefits of pair programming (Erdogmus and Williams 2003) and the importance of colocation (Bradner and Mark 2002).

5 THE RELATIONSHIP BETWEEN METHODOLOGY AND TECHNIQUE

Some queries executed against our data set seemed to suggest that the use of certain techniques among those respondents claiming to be using agile methods seemed to be very

Table 7. Actual Use of Seven Core XP Techniques Among the Sample Claiming to Follow XP

Agile Technique Used with XP	Number	Percentage of Sample
Active stakeholder participation	114	27.14%
Code refactoring	269	64.05%
Code regression testing	210	50.00%
Colocation	66	15.71%
Continuous integration	176	41.90%
Pair programming	183	43.57%
Test Driven Design (TDD)	180	42.86%

low. For example, test driven design, which from the agile methods literature one might expect to be a fundamental part of an agile approach, was only reported by between 40 and 50 percent of respondents, regardless of their chosen methods. This led us to explore in more detail the relationship between stated use of an agile method and actual use of agile techniques.

We decided to address this relationship by focusing on those respondents who claimed to be using eXtreme Programming as their agile method. There were two reasons for this. First, XP was the most popular agile method in the survey, with 23.4 percent claiming to be using XP. Second, our own analysis identified that XP appeared to be the most effective method, coupled with Scrum. Not all of the XP techniques specified by Beck and Andres (2005) were included in the original survey. Nevertheless, it would be reasonable to assume that those practitioners who claimed to be using XP would be using the core XP practices that were included in the survey. These practices would be active stakeholder participation, code refactoring, code regression testing, colocation, continuous integration, pair programming and test driven design, as shown in Table 7.

The sample size for this table was 420, which was the number of respondents who claimed to be using XP and no other method. Of these, only eight were using all of these techniques, and no single technique was being used by more than 65 percent of the sample. This result is somewhat surprising, suggesting that claiming to be doing a methodology did not necessarily mean that one was, in fact, following anything like the full set of techniques of that methodology. This seemed to go beyond the expected effects of adaptivity, and suggested an unreasonably low take up of some techniques. The obvious question that followed from this analysis was, what kind of effect might this limited use of core techniques have on the outcomes from using this method? We therefore chose to analyze which techniques might be the most important, given that many practitioners were using a subset of those recommended by the method. Our results are shown in Table 8. In order to gain a clearer result, in this analysis we combined the two techniques that focus on collaborative working, namely active stakeholder participation and colocation.

We applied a log-linear analysis to the data set for XP users to identify the associations between the techniques and their outcomes. The techniques that are ticked in Table 8 are those that have a significant association with the three performance-related

Table 8. Techniques That Appear to Show the Most Benefit in the Context of XP

Agile Technique Used with XP	Productivity	Quality	Satisfaction
Collaborative working	✓ (2)	✓ (2)	
Code refactoring	✓ (1)	✓ (1)	✓ (2)
Code regression testing		✓ (3)	
Continuous integration			
Pair programming	✓ (3)		
Test Driven Design (TDD)		✓ (4)	✓ (1)

outcomes. For each outcome, the number in parentheses shows the relative importance of that technique. For example, in the context of satisfaction, test driven design is the most important technique. Taking the three outcomes together, code refactoring appears to be the most important technique. In the context of XP, which is a code centric method, the importance of test driven design and refactoring can be understood as being crucial components of maintaining design integrity. The importance of collaborative working (in particular the XP practice of “real customer involvement”) is also underlined by our analysis.

Our final analysis addressed a further question, namely, if XP users are not using all, or most, of the available techniques, does this have a negative effect on outcomes? For this analysis, we compared the outcomes from XP users based on the number of XP techniques (between zero and seven) that they had adopted. The results of this comparison are shown in Figure 2.

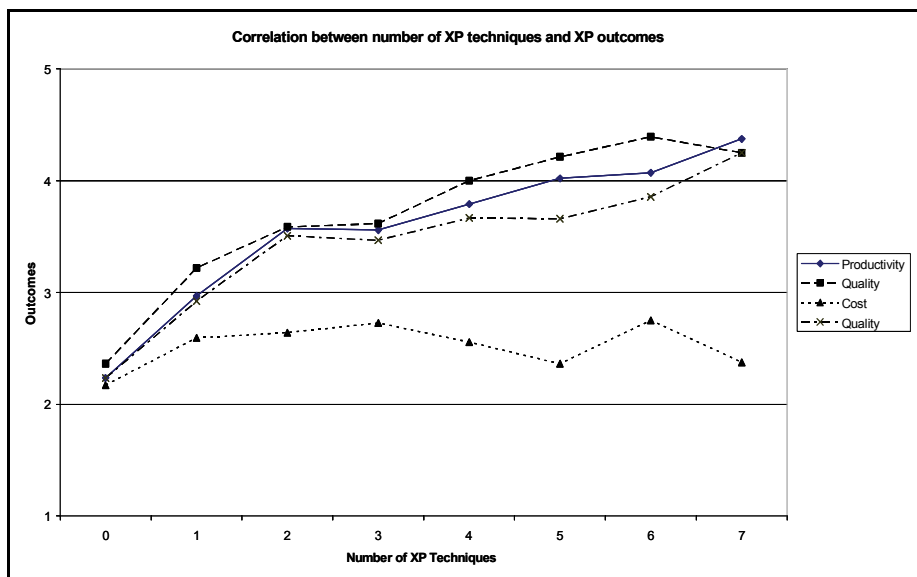


Figure 2. Graph Showing the Correlation between the Number of XP

Techniques Used and the Outcomes from Using the XP Method

As we have already identified in earlier analyses, the effect on cost seems to be independent of the number of techniques adopted, but the other three performance-related outcomes show that the more techniques that are adopted, the better the resulting performance. Those respondents who claimed to be using XP but in fact were not using any of the seven techniques had particularly poor results, which is unsurprising.

This data was further analyzed using the same classification analysis employed previously, revealing that adopting more than five techniques results in the best performance in terms of all of the measures.

6 SUMMARY AND CONCLUSIONS

The adoption of agile methods appears from some of the literature to be an untidy process of partial adoptions and adaptivity. Against this background it is helpful to try to understand which agile methods and techniques may offer the best return on investment. In this paper, we have undertaken a statistical analysis of a data set based on an on-line survey about the adoption of agile methods. From our analyses, we have drawn a number of conclusions. We have shown that adoption of at least one agile method improves the outcomes of quality, satisfaction, and productivity over the use of non-agile methods, without a statistically significant increase in cost. We have also shown that the most effective way to apply agile methods is to combine more than one method together, and the most effective combination of methods appears to be eXtreme Programming and Scrum. We also looked at agile techniques and their outcomes and identified pair programming and colocation as the two most significant techniques when analyzed across all agile methods. However when we concentrated on an analysis of XP, the most important techniques for this method appeared to be code refactoring, collaborative working (colocation and active stakeholder involvement), and test driven design. Finally, we showed that in order to gain the full benefits from adopting the XP method, at least five of the core XP techniques had to be used.

From the work we have undertaken, we make a number of proposals for software development teams using or migrating to agile methods. First, it appears that successful adoption of an agile approach does not necessarily just mean selecting an individual method. Rather, it may be better to consider blending multiple complementary methods. Second, although it is an acceptable practice to adapt methods by selecting from techniques, it is important to select those techniques that offer the best outcomes, rather than adopting only those that are easy or do not require so much effort to integrate into existing processes. Third, it is important to recognize that, although not all techniques of an agile method are compulsory, there will always be a critical mass of techniques that should be adopted in order to offer the best chance of project success. Finally, given the insights that the data used in this paper has provided, it would be useful for development teams to monitor the effectiveness of their agile project practices by gathering data on the outcomes of quality, productivity, satisfaction, and cost on an ongoing basis, enabling them to carry out their own project metrics.

There are a number of issues with this analysis that should be borne in mind when considering our results. Because this was an on-line survey, the respondents were self selecting. We cannot, therefore, guarantee the validity of their responses. We are also

unable to determine from the data whether respondents are reporting the use of multiple methods because different teams within their organization use different methods, or because individual teams are combining methods. We have also focused in this paper on agile methods and techniques. However the original survey includes questions about skill level and organization size, which Ambler has addressed separately (Ambler 2006c) but which we have not attempted to include in this analysis. In addition, further information about the respondents such as what type of organization they work for and what type of software systems they are developing is not available to us.

Ambler has himself summarized a number of issues with the survey (Ambler 2006b), including potential misunderstandings by respondents about feature driven development (FDD), which may make answers relating to this method unreliable, and the absence of the rational unified process (RUP) from the specified list of methods. This method appears a number of times in the “other” category within the survey but has not been discriminated in the analysis.

The results of our analysis appear to indicate a number of areas for future work. In particular, our quantitative analysis suggests a number of areas where field studies and qualitative analysis might be undertaken to further investigate issues such as how software development teams select, combine, and adapt agile methods in practice, and why particular subsets of techniques are selected from agile methods. There may also be scope for a further survey that might attempt to provide a finer grained discrimination of questions related to method and technique so that we might identify the ways that multiple agile methods are being used in practice. Finally, the original survey results are still available for public download (Ambler 2006a) and there are further aspects of the data, not considered in this paper, that could be analyzed from new perspectives.

References

- Abrahamsson, P., Salo, O., Ronkainen, J., and Warsta, J. *Agile Software Development Methods: Review and Analysis*, Oulu, Finland: VTT Publications, 2002.
- Ambler, S. “Agile Adoption Rate Survey,” *Amblysoft*, March 2006a (available online at <http://www.amblysoft.com/surveys/agileMarch2006.html>).
- Ambler, S. “Agile Adoption Rate Survey: Discussion of the Results,” *Amblysoft*, March 2006b (available online at <http://www.amblysoft.com/surveys/agileMarch2006.html#Discussion>).
- Ambler, S. “Survey Says: Agile Works in Practice,” *Software Development Magazine*, August 3, 2006c (available online at <http://www.ddj.com/dept/architect/191800169>).
- Anderson, J. D. *Agile Management for Software Engineering: Applying the Theory of Constraints for Business Results*, Upper Saddle River, NJ: Prentice Hall, 2004.
- Aveling, B. “XP Lite Considered Harmful?,” in J. Eckstein and H. Baumeister (eds.), *Extreme Programming and Agile Processes in Software Engineering: 5th International Conference*, Garmish-Partenkirchen, Germany, June 6-10, 2004, pp. 94-103.
- Beck, K., and Andres, C. *Extreme Programming Explained: Embrace Change* (2nd ed.), Boston: Addison-Wesley, 2005.
- Beedle, M. “Agile Enterprise,” 2006 (available online at <http://www.e-architects.com/AE/>).
- Bhat, T., and Nagappan, N. “Evaluating the Efficacy of Test-Driven Development: Industrial Case Studies,” in *Proceedings of the 2006 ACM/IEEE International Symposium on Empirical Software Engineering*, Rio de Janeiro, Brazil, September 21-22, 2006, pp. 356-363.
- Boehm, B. “Get Ready for Agile Methods, with Care,” *IEEE Computer* (35:1), 2002, pp. 64-69.

- Bradner, E., and Mark, G. "Why Distance Matters: Effects on Cooperation, Persuasion and Deception," in *Proceedings of the 2002 ACM Conference on Computer-Supported Cooperative Work*, New Orleans, LA, 2002, pp. 226-235.
- Canfora, G., Cimitile, A., Garcia, F., Piattini, M., and Visaggio, C. "Evaluating Advantages of Test Driven Development: A Controlled Experiment with Professionals," in *Proceedings of the 2006 ACM/IEEE International Symposium on Empirical Software Engineering*, Rio de Janeiro, Brazil, September 21-22, 2006, pp. 364-371.
- Datta, S. "Agility Measurement Index: A Metric for the Crossroads of Software Development Methodologies," in *Proceedings of the 44th Annual ACM Southeast Regional Conference*, Melbourne, FL, 2006, pp. 271-273.
- Dominguez, J., Linecar, P., and Black, S. "Visualization of a Suitability Filter for Agile Methods," in R. Dawson, E. Georgiadou, P. Linecar, M. Ross, and S. Staples (eds.), *Software Quality Management XIV: Perspectives in Software Quality, Proceedings of the 14th International Software Quality Management Conference*, Southampton, UK, April 10-12, 2006, pp. 299-319.
- Dustin, E., Raskha, J., and Paul, J. *Automated Software Testing*, Reading, MA: Addison Wesley, 1999.
- El Emam, K. "Finding Success in Small Software Projects," *Agile Project Management Executive Report* (4:11), 2003.
- Erdogmus, H., and Williams, L. "An Economic Analysis of Pair Programming," in L. Williams and R. Kessler (eds.), *Pair Programming Illuminated*, Boston: Addison-Wesley, 2003, pp. 221-236.
- George, B., and Williams, L. "An Initial Investigation of Test-driven Development in Industry," in *Proceedings of the ACM Symposium on Applied Computing*, Melbourne, FL, March 9-12, 2003, pp. 1135-1139.
- Grossman, F., Bergin, J., Leip, D., Merritt, S., and Gotel, O. "One XP Experience: Introducing Agile (XP) Software Development into a Culture That Is Willing but Not Ready," in H. Lutfiyya, J. Singer, and D. Stewart (eds.), *Proceedings of the 2004 Conference of the Centre for Advanced Studies on Collaborative Research*, Markham, Ontario, Canada, October 4-7, 2003, pp. 242-254.
- Highsmith, J. *Agile Software Development Ecosystems*, Boston: Addison-Wesley, 2002.
- Hussman, D. "A Fishbowl with Piranhas: Coalescence, Convergence or Divergence? The Future of Agile Software Development Practices: Some Assembly Required," in *Proceedings of the Conference on Object Oriented Programming Systems Languages and Applications*, Portland, Oregon, October 22-26, 2006, pp. 937-939.
- Keenan, F. "Agile Process Tailoring and Problem Analysis," in *Proceedings of the 26th International Conference on Software Engineering*, Edinburgh, UK, May 23-28, 2004, pp. 45-47.
- Lindvall, M., Muthig, D., Dagnino, A., Wallin, C., Stupperich, M., Kiefer, D., May, J., and Kahkonen, T. "Agile Software Development in Large Organizations," *IEEE Computer* (37:12), December 2004, pp. 26-34.
- Mar, K., and Schwaber, K. "Scrum with XP," *Informit.com*, March 22, 2002 (available online at <http://www.informit.com/articles/article.asp?p=26057&rl=1>; article provided courtesy of Prentice-Hall).
- Mišić, V. "Perceptions of Extreme Programming: An Exploratory Study," *ACM SIGSOFT Software Engineering Notes* (31:2), 2006, pp. 1-9.
- Muller, M. M., and Hagner, O. "Experiment About Test-First Programming," *IEE Proceedings Software* (149:5), 2002, pp. 131-136.
- Nerur, S., Mahapatra, R., and Mangalaraj, G. "Challenges of Migrating to Agile Methodologies," *Communications of the ACM* (48:5), 2005, pp. 73-78.
- Paige, R., Chivers, H., McDernid, A., and Stephenson, Z. "High-Integrity Extreme Programming," in *Proceedings of the ACM Symposium on Applied Computing*, Santa Fe, New Mexico, March 13-17, 2005, pp. 1518-1523.

- Sfetsos, P., Angelis, L., and Stamelos, I. "Investigating the Extreme Programming System: An Empirical Study," *Empirical Software Engineering* (11:2), 2006, pp. 269-301.
- Shine Technologies. "Agile Methodologies Survey Results," Shine Technologies Pty. Ltd., 2003 (available online at <http://agilealliancebeta.org/system/article/file/1121/file.pdf>).
- Stephens, M., and Rosenberg, D. *Extreme Programming Refactored: The Case Against XP*, New York: Apress, 2003.
- Williams, L., Maximilien, M., and Vouk, M. "Test-Driven Development as Defect-Reduction Practice," in *Proceedings of the 14th International Symposium on Software Reliability Engineering*, Denver, CO, November 17-21, 2003, pp. 3-4.

About the Authors

David Parsons is a senior lecturer in Information Systems at Massey University, Auckland, New Zealand. Formerly an Enterprise Java consultant in the U.K., his research interests include agile methods, web application architectures and mobile learning. He can be reached by e-mail at d.p.parsons@massey.ac.nz.

Hokyoung Ryu is a lecturer in Information Systems at Massey University, Auckland, New Zealand. He is active in research on how new information and communication technologies such as interactive TV and mobile systems will change human social behavior. He can be reached by e-mail at h.ryu@massey.ac.nz.

Ramesh Lal is a researcher in Information Systems at Massey University, Auckland, New Zealand. He is currently working for his Ph.D., studying the adaptivity of agile methods using case studies across Australasia. He can be reached by e-mail at r.lal@massey.ac.nz.