Usage and Perceptions of Agile Software Development in an Industrial Context: An Exploratory Study

Andrew Begel
Microsoft Research
One Microsoft Way
Redmond, WA 98052
andrew.begel@microsoft.com

Nachiappan Nagappan
Microsoft Research
One Microsoft Way
Redmond, WA 98052
nachin@microsoft.com

Abstract

Agile development methodologies have been gaining acceptance in the mainstream software development community. While there are numerous studies of Agile development in academic and educational settings, there has been little detailed reporting of the usage, penetration and success of Agile methodologies in traditional, professional software development organizations. We report on the results of an empirical study conducted at Microsoft to learn about Agile development and its perception by people in development, testing, and management. We found that one-third of the study respondents use Agile methodologies to varying degrees, and most view it favorably due to improved communication between team members, quick releases and the increased flexibility of Agile designs. The Scrum variant of Agile methodologies is by far the most popular at Microsoft. Our findings also indicate that developers are most worried about scaling Agile to larger projects (greater than twenty members), attending too many meetings and the coordinating Agile and non-Agile teams.

1. Introduction

Agile software development (ASD) methodologies [7] have been gaining acceptance among mainstream software developers since the late 1990s, when they were first postulated in the forms of Scrum [14], Crystal [8], Extreme Programming [4] and other methodologies. Today they are established to varying degrees in the academic, educational and professional software development communities.

We would like to understand how ASD methodologies are used, what kind of acceptance and spread they have, and what kind of successes and failures occur in each of these communities. We believe strongly in using empirical methods to explore questions engendered by these research topics. While there is much to be learned from looking at the software artifacts created by developers and from measuring developer productivity and software failure proneness, we can gain great insights through direct interaction with software developers. We can learn about their development practices, their perceptions of development processes, and how the two interact.

We conducted a web-based survey of Microsoft employees in development, testing and management roles who are directly involved in the production of software. Our questions were targeted to understand respondents' demographics, ASD usage, penetration of ASD practices, and their perceptions of why ASD works well or poorly on their software teams. We received a response rate of 17%; the nearly 500 responses make it one of the largest respondent populations for a survey of software development at Microsoft. From these responses, we gained a fairly clear picture of how ASD is used at Microsoft.

Our findings indicate that around one-third of the respondents use ASD. Scrum is the most popular ASD methodology. ASD is a relatively new phenomenon to Microsoft; most projects have employed Agile for less than two years. ASD is used mostly by collocated teams who work on the same floor of the same building. Finally, ASD users have an overwhelmingly positive opinion about it.

The rest of this paper is organized as follows. In Section 2 we discuss our contributions and in Section 3 review the related research. Section 4 describes the experimental methodology and illustrates the results. In Section 5, we discuss the benefits and problems of ASD as perceived at Microsoft. Section 6 concludes with a review of our most important findings and their implications for future research here at Microsoft and at other sites.

2. Contributions

ASD methodologies are becoming more popular in industry and very little is understood about their penetration, benefits and problems [16]. We find the lack of such reported results surprising. Our main contributions to the current state of the art are:

- *Large scale industrial view of ASD:* Our study is one of the first performed on a large scale (487 respondents) in an organization to assess the extent to which groups use ASD.
- *Dominant Agile practices and methodologies:* We identify the most commonly used Agile practices, and indicate the practices that have not gained acceptance within our development community. In addition, we identify the commonly used ASD methodologies (Scrum, XP, Crystal, etc) within Microsoft.
- *Common benefits and problems associated with ASD*: We report on issues related to Agile adoption and practice observed by the software engineers at Microsoft.

In general, though our results are specific to Microsoft, the results provide valuable insight into how software organizations are adopting and adapting ASD. We hope to identify collaborators in other software organizations and people in academia to replicate this study in different contexts to write a comprehensive report on the penetration of ASD in the software engineering community.

3. Related Work

ASD is the topic of much debate at several software development organizations. Unfortunately there is little or no evidence as to the extent to which ASD is used in large commercial software development organizations, nor the practices that are followed, nor people's perception towards ASD practices, etc. Our paper aims to address these limitations to some degree by assessing the current state of the practice at Microsoft on the usage, penetration and success of ASD.

Abrahamsson et al. [1] demonstrated how to collect metrics to measure productivity, quality and schedule estimation for an ASD project using XP. Williams et al. [17] investigated the usage of a subset of XP [5] practices at a group in IBM. The product developed at IBM using XP was found to have significantly better pre-release and post-release quality compared to an older release. The teams using XP reported an improvement in productivity and morale. In addition, customers were more satisfied with the product developed using XP because the teams delivered more than what the customers had originally asked for. Similar results were obtained for a case study conducted at Sabre airline systems [10]. Maurer et al. [11] studied the development of a web based system by nine full time employees in a small company that used XP and observed substantial productivity gains compared to their pre-XP timeframe.

Our work is closely related to the work by Melnik and Maurer [13]. They investigated the perception of students towards ASD by collecting qualitative and quantitative data over three academic years. Overall, the students were positive towards using XP. A serious limitation to their experiment was the relative naiveté of student perceptions of XP. For example, students indicated that their productivity increased using XP. But they might have used a beginning programming course (CS1) as a yardstick for comparison which can skew the results in favor of XP. Students are similarly naïve about software quality. Melnik and Maurer's study also reports several weak correlations to draw relationships between several survey parameters (correlation less than $\pm$ 0.5). Carver et al. [6] discuss using students as subjects in empirical studies. Academic case studies do provide a meaningful ground for researchers to try out ideas before replicating them in industry. From an industry perspective there has been limited empirical evidence on the usage/perception towards ASD practices. Sharp and Robinson present an overall ethnographic picture of XP practices in a small company [15], but other work has only addressed individual practices such as pair programming [2] and test-driven development [12].

4. Experimental methodology

Our research was conducted using a anonymous web-based survey offered over a period of two weeks in October 2006. An invitation was sent by email to 2,821 recipients, randomly selected from a much larger pool of around 28,000 software developers, test developers, and managers (a 10% sample was selected). We received 492 responses, of which 4 were invalid (for technical reasons), for an overall response rate of 17%. Response rate for developers was 18%, testers were 18%, and managers were 10%. Respondents could identify themselves (separate from their survey responses) to enter a drawing for a $250 reward.

Respondents were asked a total of 46 questions divided into three sections: demographics, Agile development, and pair programming. We will report on the results of the first two sections, illustrating what we discovered about Agile development (only 6% of respondents reported using pair programming on their current team). In the Agile development section, we asked whether the respondent had ever used Agile, whether they used it in their current team at Microsoft, and which forms of Agile they used. We also asked which of the common ASD practices they used (or might use in the future) in their team. Responses for this question were Yes, Sometimes, No, Planning To, and Never. We then asked a series of questions to find out their overall impressions of Agile development, whether they liked it, and whether it was better than any previous method they had used for collaboration, coordination and morale. We then asked all survey respondents whether they liked Agile and what they thought its top three benefits and problems were, regardless if they had used or did not use ASD. This section of the survey was free-response. We ended by asking if anyone had used Agile in the past but did not now, and why they stopped using it.

All of the free response answers were printed out on a few thousand note cards. We organized the cards using a card sort designed to categorize the responses by thematic similarity (as illustrated in LaToza et al.'s earlier survey at Microsoft [9]). The themes that emerged during the sort were not chosen beforehand. Respondents reported 687 Agile benefits, for which there were 44 common themes. 565 problems were reported, and grouped into 58 themes.

5. Experimental results

In this section, we report on the findings from the survey demographics and the free response perceptions of ASD.

5.1 Demographics

Respondents had an average of 9.20 years experience in the software profession (standard deviation was 7.06; minimum 0 years; maximum 35 years). They worked on their current team for an average of 2.4 years (standard deviation 2.5). The respondents were spread across different geographical locations in North America, Asia and Europe. Of all our respondents, 72.6% were individual contributors, 16.5% were managers and 7.2% were managers of managers.

Our demographics indicate that our survey respondents were fairly experienced and had spent more than two years on their current team. They understood their development practices well enough to provide a relative assessment. Also the distribution of our respondents across North America, Asia and Europe increases the diversity of our responses by providing a global perspective.

5.2 Extent of adoption

From an adoption standpoint we observed that 156 out of the 487 people said that their team used ASD. This is significantly higher than what was believed to have been known / perceived at Microsoft about the penetration of ASD practices. Table 1 provides the raw data. Further, we observe that among teams 59.6% of the people who use Agile methodologies work on legacy products, i.e. not a version 1 product. This is contrary to popular opinion that ASD is not used in legacy systems.

Table 1: Adoption of ASD

Does your team use Agile Methodology?				
		No	Yes	Total
Is this Version 1 of your product?	n/a	45	7	52
	No	204	**93**	297
	Yes	72	55	127
	Total	322	**156**	478

5.3 ASD methodologies

There are several ASD methodologies available today. 125 out of 192 of this question's responses indicated they used the Scrum [14] ASD methodology. Figure 1 shows the extent of adoption of different ASD methodologies. The respondents who answered *other* (22) were asked to specify the ASD process they used. Most mention a variant of Scrum or a practice loosely based on Scrum.

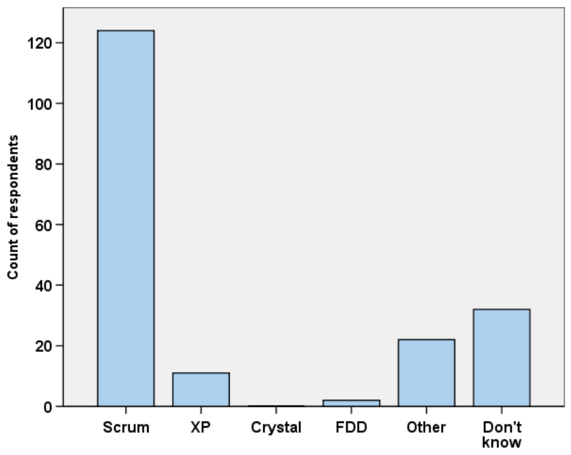

Figure 1: Different ASD Methodologies

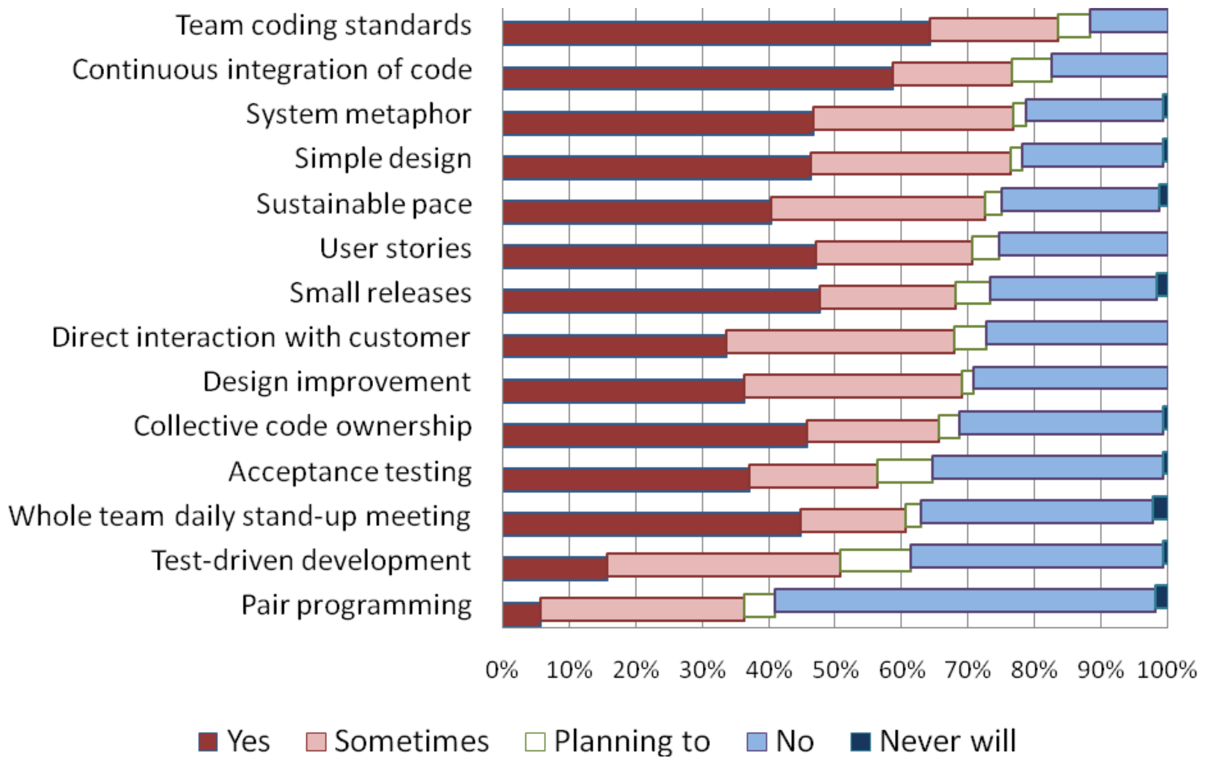

Figure 2: Percentage of Usage for Agile Practices

5.4 Agile practices used

We asked survey respondents to tell us the extent of their usage of various Agile practices. Figure 2 represents the responses sorted in order of greatest use (Yes + Sometimes + Planning to) among the practices. The top Agile practices that teams followed were team coding standards and continuous integration of code. The least followed practices were pair programming and test driven development (TDD). All the practices except pair programming had at least greater than 60% (current and planned) adoption.

5.5 Time of adoption

We analyzed how long ASD has been used at Microsoft in order to identify how recent a trend is ASD at Microsoft. Figure 3 shows the distribution of the number of months ASD has been practiced in the respondents' teams. The figure indicates that the length of time is skewed heavily towards zero. More than 90% of the projects have used Agile for less than 24 months (2 years). The average time extent is 8.3 months (standard deviation is 8.9).

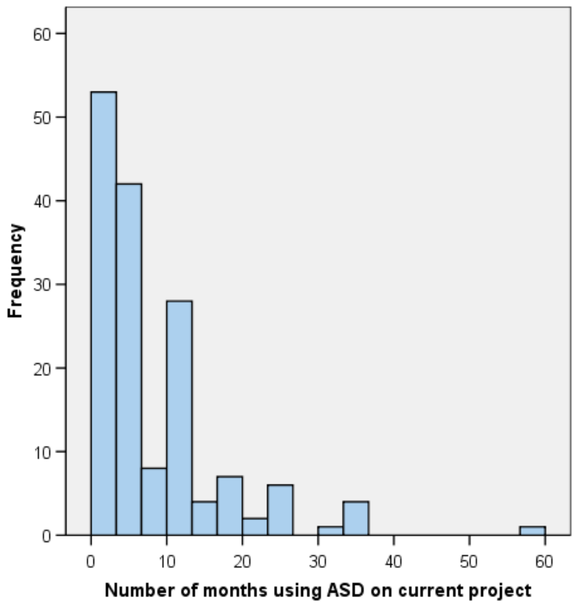

Figure 3: Length of Time of ASD Usage

5.6 Are Agile teams collocated?

In Microsoft a fair amount of development takes place in multiple locations (both physical and geographical). An important question for software development organizations is the location of teams adopting ASD methodologies. From Table 2 we observe that among the teams practicing Agile software development, collocated teams (same office, hallway, floor, or building) account for more than 83.9% (shown in bold) of the respondents. Very few teams distributed across cities and countries use ASD methodologies. The demographics for the teams that do not use ASD is also shown in Table 2 for comparison. An interesting point to note is that teams are collocated regardless of which software methodology they use (83.4% - shown in italics)).

Table 2: ASD Collocation Data

	Use Agile?	
Collocated dynamics	Yes	No
Not-collocated	12	29
Same country	4	10
Same city	1	2
Same campus	8	12
Same building	**24**	*35*
Same floor	**68**	*138*
Same hallway	**25**	*76*
Same office	**14**	*18*
Total	*156*	*321*

5.7 Individual attitudes towards ASD

We then asked participants whether they liked ASD. Figure 4 shows the results. Among people who currently use Agile (left side of the graph), 89.7% like or are neutral to ASD. A more important point is that among groups that do not use ASD, 92.8% said they liked or were neutral to ASD, indicating that a vast majority of developers are open to trying ASD in the future. From an analysis perspective we correlated the age of the respondents to their inclination to like/dislike ASD. There was no correlation between the factors, indicating that age was not related to people's like or dislike of Agile. Similar results were obtained correlating the collocation information with the inclination of developers to like or dislike ASD.

5.8 Team attitudes and morale factors

Figure 4 also shows team attitudes and morale concerning ASD. Around 60% of the respondents agree that ASD is working well for them and their team. Less than 40% however, agree that ASD is working well for the larger group. We discuss some reasons why in Section 7. Less than 20% of the respondents say that their team morale has decreased due to ASD, and only 10% mention that ASD has affected communication (negatively) between the development and test teams.

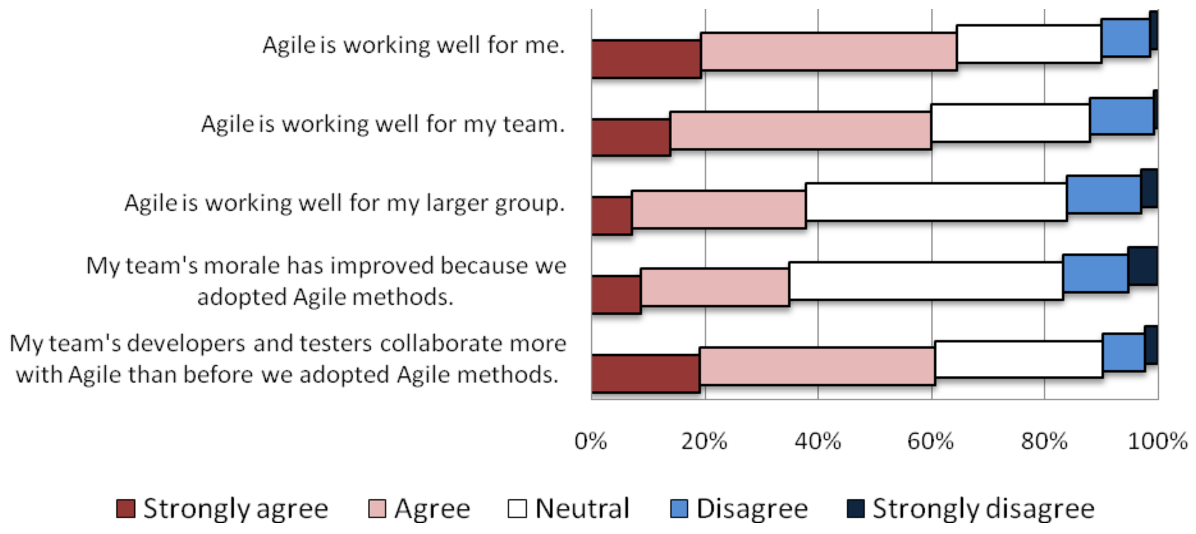

Figure 4: Team Attitudes and Morale Factors Concerning ASD

5.9 ASD Benefits

We asked the survey participants what they thought were the top three benefits and problems with Agile Development. Comments from the respondents are presented in italicized form to add more contextual information as appropriate for our discussion. Table 3 presents the top 10 problems of ASD as perceived by the respondents and the number who cited it as a benefit.

Table 3: Benefits to Agile Development Methodologies

1. Improved Communication and Coordination	121
2. Quick Releases	101
3. Flexibility of Design – Quicker Response to Changes	86
4. More Reasonable Process	65
5. Increased Quality	62
6. Better Customer Focus	50
7. Improved Focus -- Better Prioritization	28
8. Increased Productivity	26
9. Better Morale	23
10. Testing First	22

The top benefit was improved communication and coordination among team members. Specifically, the daily scrums were seen as instrumental, and were especially useful to bring testers and developers together. Improved awareness of team members' activities was another benefit. One respondent said "*Team members are aware of what each of the others is working on*." Another promoted the benefits of earlier discovery and handling of development issues: "Better overall communication (quicker discovery of problems, etc)."

The second most cited benefit was Quick Releases. This was a consequence of Continuous Integration, a feature of Extreme Programming. Developers create demo-able releases every few weeks instead of every few months or years. This makes it easier to keep track of progress and monitor software quality, as one respondent said, "*Software functionality progress can be checked and monitored much more frequently rather than at end of long milestones*." It makes it easier to evaluate the value of features and the product, provides feedback to improve the product, and improves turnaround time for fixing bad bugs. A tester commented, "*When you integrate early and often, the product can be tested early and often, too*."

In third place is Flexibility of Design. Developers noted that short sprints combined with more emphasis on customer feedback led to better agility and efficiency in responding to changing requirements, internal processes, reorganizations or politics, and flushed out bad designs more quickly. "*You don't have to commit prematurely (for example, to design decisions)*." An Agile process "*anticipates changes to requirements so that they do not destroy a schedule*." Flexibility was not solely based on the product, but the development process itself, i.e. "*Ability to change directions quickly i.e. cancel a sprint and start another*." Another said "*quick results lead to iteration which helps us to fail cheaply instead of in an expensive way (if we fail).*" "*Agile embraces change, which is a fact and part of software development*."

The fourth most popular benefit of ASD is a More Reasonable Process. Many developers complained about rigid development processes that were relaxed in an Agile environment. Developers wasted less time on tasks they perceived as irrelevant, such as "*large specs that are out of date before they are finished*." Some of this perception may be a reaction to earlier more waterfall-like processes used at Microsoft, but some may come from the haphazard adoption of Agile methodologies by various groups, very few of which would characterize themselves as Agile experts. Documentation and planning are viewed as just-in-time and just-enough for the next sprint. One developer notes that "*process improvement is built into the process*." A program manager said that running the process is more manageable and less bureaucratic than earlier processes. The process supports "*real-time tracking of progress and ability to adjust future forecasts based on real data*." Agile methodologies are more dynamic and incur less overhead. One manager said the Agile process costs less.

Not far behind was Improved Quality. The quality of the software is a strong concern of developers. The effects were manifested as fewer bugs, and a more stable set of features. Test-driven development and test automation were seen as factors that contributed to higher code quality. All aspects of software are improved, from design and architecture to performance of the products of each sprint. Improved communication leads to faster turnaround time for blocking bugs. One developer said "*ongoing refactoring leads to higher code reuse and better quality.*"

Rounding out the top ten benefits of Agile development were better focus on customers, better prioritization of development and focus on the product, improved productivity, increased morale (often tied to continuous integration with deliverables at the end of each sprint), and more reliance on test-driven development. The Appendix at the end of the paper lists the other perceived benefits of ASD from a completeness perspective.

5.10 ASD Problems

Many developers perceived problems with Agile development processes. Table 4 highlights the top 10 problems with ASD as perceived by the respondents and the number who cited it as a problem.

Table 4: Problems with Agile Development Methodologies

1. Does not scale to larger projects	52
2. Too many meetings	44
3. Management Buy-in	37
4. Unfamiliar with Agile	36
5. Coordination with other teams	29
6. Lose sight of big picture	29
7. Culture	27
8. No up-front design, bad design	23
9. Lack of schedule	19
10. Dev/Test Integration is Difficult	19

The top concern of developers at Microsoft with Agile development is whether these methods scale to larger software teams. We did not gather data on team size of those who said they used Agile development, but our data does show that most Agile teams are on the same hallway or floor of a building, limiting group size to 30 people or so. Several people on the survey expressed concern about scaling to group sizes of several hundred or even higher. One developer said that Agile "*works for small co-located teams, but not for complex large projects*." It can be difficult for larger teams to be as flexible as smaller teams with respect to design and architectural changes. Scrum meetings were sometimes considered inefficient, especially when the team was inexperienced with Agile, or it was large (over 8-10 people). Apprehensions concerning scaling to products with long release cycles or large legacy codebases were also mentioned.

The second concern was about the Scrum. Scrum meetings involve all members of a team and often occur daily. Many respondents complained about the inefficiency of these meetings, especially when they were poorly run by a Scrum Master who was not disciplined and focused enough to run the meeting quickly. We did not correlate whether respondents who said that Scrums take too long also said whether they get anything positive from the Scrum. Meetings were also viewed as ways for managers to micromanage their teams: "*what have you done in the last 24 hours?*" Some developers were uncomfortable reporting their progress: "*Personally for me, the daily standup scrum meetings were more distracting than helpful. The pressure to daily report percentage of progress was uncomfortable, especially when I had to report progress (or call an item 'done') without actually testing in integrated fashion*."

Management buy-in was the third concern. Many program managers were worried that upper-level management would ask for progress reports and productivity metrics that would be hard to gather in an Agile work environment. Management ignorance of Agile methodologies was also a worry. Will Agile advantages be able to overcome the well-known existing problems in software development? Management sometimes worries that not all development teams are cut out for Agile development. The flexibility in scheduling afforded by sprints is unfamiliar to managers used to hearing about feature milestones planned at the beginning of a project. In addition, because it is easy to move features to later sprints as work piles up, it is not easy to predict when a particular feature will go into the product. "*Upper management still tries to get specific dates for specific deliverables*." This scheduling difficulty is combined with problems estimating the cost of a project before it starts. Sometimes management tries to covertly switch back to a Waterfall model. They "*want to use an Agile development process that is in fact a more 'classic' engineering process or are extremely date-driven combined with 'gotta-have' deliverables*."

The fourth concern was an apprehension about learning Agile development. Some developers wished they had formal training to do Agile, noting that there were few training options available to them. Many who commented on training appeared to have the idea that if they did not do Agile *perfectly* then the product or process would suffer. There was also some concern that some developers are not cut out for Agile: Agile "*requires some interpersonal skills which may not be abundant in the IT sector.*" Social issues also play a role here as social cliques may form and become the dominant means of communication between group members. Those not in the clique may miss out on important communications. Agile development "*is simple, but requires a lot of discipline from the team*." Cultural issues also play into adoption of Agile development. "*We are not patient enough to make a plan, do the design work, then make a schedule, and then execute*." Agile development often requires a change in mindset that developers may not be eager to undertake. Several developers also note that unless there is full adoption by the team, Agile methodologies do not work very well.

The fifth concern is about coordinating with other teams. This is especially worrisome in larger projects where only a few groups are Agile and the rest are using a typical Waterfall model. Problems arise in the scheduling of deliverables between dependent projects. Non-Agile groups deliver builds only when they're fully coded and tested, and these milestones are few and far between. Agile groups often schedule features to be complete during each sprint and may race ahead of their non-Agile peers. There is a perception that non-Agile groups do not understand the scheduling requirements that Agile groups use and those that they require of their dependencies. "*Interaction with non-Agile teams is hard because they don't understand that you can guarantee that all the sprint items will be completed because the prioritization meeting involves very loose time estimates.*" Also, as the number of modules in the project grows larger, respondents noted more difficulty, cost, and time integrating modules into the whole.

Losing sight of the big picture rounds out the top six concerns with Agile development. This is because "*you're so focused on the day to day deliverables*." The "*focus is on today's work*" more "*than what the feature team is trying to achieve.*" In addition, development items that require more time than a single sprint or never rise to high priority can get completely forgotten. One developer would like to see a kind of "*forcing function for envisioning process beyond one or two sprints*." Another notes that it is difficult concentrate on the design properly; instead of getting it right in the first place, teams rely on design improvements as they go.

The rest of the top ten problems include culture clashes during Agile adoption, lack of a precise and overarching design before the project starts, the lack of a fixed and predictable schedule, and a perceived difficulty integrating developers and testers. The Appendix at the end of the paper lists the other perceived problems of ASD that had a lower frequency of occurrence in the problems list.

5.11 ASD Attrition

Analyzing people who no longer use ASD, 53 developers had used ASD in the past but no longer use it on their current project. They provided a variety of reasons. Table 5 lists these reasons and the count of the respondents who cited them.

Table 5: If you used Agile in the past, but do not now, why?

1.	Changed jobs; new team doesn't use Agile	23
2.	Not coding anymore	3
3.	Management in the way	3
4.	Practice something similar to Agile	3
5.	Tried to introduce Agile, but failed	3
6.	Hard to coordinate with non-Agile teams	2
7.	Agile projects suffer from poor design	2
8.	Used Agile in school but not anymore	2

The most common reason was that they had switched jobs and their current team no longer used Agile. Some were no longer programming, thus Agile did not make sense. A couple had used Agile development in school, but did not join an Agile development group. A few used practices that were similar to Agile, such as buddy reviews instead of pair programming and unit testing instead of TDD. Several tried to introduce Agile development to their team, but failed either in management buy-in or in the actual implementation of Agile, so it was abandoned. Some others expressed concerns similar to those reported above about the problems with Agile development. These included the difficulty coordinating with other non-Agile teams, and the lack of good design for Agile projects. Other lesser-reported reasons were difficulties starting Agile due to workload, too much legacy code, too much documentation to write or to not finding enough people who could adapt to Agile.

6. Threats to Validity

From an internal validity point of view the study was conducted by two researchers at Microsoft Research. Microsoft Research is a parallel organization when compared to the Microsoft product groups. Neither of the two authors, nor any of the respondents shares any common management nor are they part of the same management chain. The survey was conducted anonymously, and there was no necessity for the respondents to answer for/against ASD. The benefits and problems were self-reported in free-form to remove any bias that could have been introduced by the authors asking the respondents to pick the benefits and problems of ASD from a list. Furthermore, the authors have no influence on the use or perception of ASD in the product groups. Another threat to internal validity would be that people practicing ASD would have been more likely to respond to a survey on ASD.

Two other threats to internal validity are on statistics for number of people using ASD on legacy systems. It is possible that this might be a reflection of the organizational layout of Microsoft/any other large software company. Additionally we do not have statistics on each individual's team size as teams can be a flexible and not necessarily reflected by the management hierarchy. We plan to address these issues in our future work.

From an external validity point of view, this study is based on only one large organization. But within that organization, our respondents are from various groups that are involved all the way from designing operating systems to games to web service applications. Drawing general conclusions from empirical studies in software engineering is difficult because any process depends to a large degree on a potentially large number of relevant context variables. For this reason, we cannot assume a priori that the results of our study generalize beyond the specific environment in which it was conducted [3]. Researchers become more confident in a theory when similar findings emerge in different contexts [3]. Towards this end we intend that our case study will be replicated in different software organizations. The future work section discusses this aspect in more detail.

7. Conclusions and future work

To summarize our main findings, around one-third of respondents are using ASD methodologies in some form. The Scrum methodology is by far the most popular, with 65% of respondents using it on their software teams. Our respondents were fairly experienced with a mean work experience in the software profession of 9.2 years. Team working on legacy systems were used ASD. Out of 14 ASD practices, 60% of respondents used 12 or more of them. The two least used practices are test-driven development and pair programming.

Respondents also told us what they liked and disliked about ASD methodologies. Most view ASD favorably due to improved communication between team members, quick releases and the flexibility of designs in the Agile process. On the other hand, developers worry about scaling Agile to larger projects (greater than 20-30 members), attending too many meetings which contribute to excessive overhead, and experiencing difficulty getting management to buy into ASD methods. Some respondents no longer use ASD, but have in the past. Usually this was due to switching jobs to a group that did not practice ASD, but was also due to difficulties getting one's new group to adopt Agile. Some tried it and failed to make it work, some had trouble with gaining management buy-in and others adapted Agile beyond recognition until they felt it would be better called something "like" Agile, but not specifically Agile. In general, there was an impression by developers that there was a one true way to practice ASD and if they were not following this way to the letter, that they were somehow doing it wrong and would engender unforeseen consequences. This is somewhat ironic since Agile methods are above all supposed to be adaptive to the needs of the project.

The results of our study contribute to our understanding of how ASD methodologies are being implemented in the workplace. Our research goals are to understand first the state of the practice at Microsoft and delve into the details from there. The follow up work for our analysis is outlined below:

- **Scaling:** In prior reported results on the use of ASD, teams have typically between 2-20 members. Teams at Microsoft and other companies can be much larger, between 500-5000. We would like to investigate how Agile can be adapted to work for these large teams.
- **Coordination:** In large companies Agile is not adopted simultaneously by all teams. We plan to study how Agile teams coordinate dependencies and deliverable with non-Agile teams.
- **Empirical body of knowledge:** Collaborate with people in the empirical community to replicate these studies in industry and academia to build an empirical body of knowledge about the various facets of ASD.
- **Product and process measurement:** Measuring product measures (LOC, complexity, failures etc.) and process measures (productivity, requirement volatility etc.) for ASD projects to compare against non-ASD projects. This would enable us to identify the proper contexts in which ASD should be used.
- **Tools and resources:** Identify areas to develop tools for ASD to improve communication, quality and scheduling and estimation.
- **Ethnographic studies:** Now that we know what methodologies are used, we can conduct interviews and study groups using ethnographic techniques to learn how to identify and alleviate some of the more specific problems they face deploying ASD.

Acknowledgements

We would like to thank the anonymous respondents at Microsoft for the feedback that made this paper possible. We would also like to thank Rob DeLine, Gina Venolia and Tom Ball for their support and feedback on this work.

Contact

Researchers interested in replicating this study should contact the authors: andrew.begel@microsoft.com, nachin@microsoft.com to obtain an editable/reusable copy of the survey.

References

[1] P. Abrahamsson, Koskela, J., "Extreme Programming: A Survey of Empirical Data from a Controlled Case Study", Proceedings of International Symposium on Empirical Software Engineering, pp. 73-82, 2004.

[2] E. Arisholm, Gallis, H.E., Dybå, T., Sjøberg, D., "Evaluating Pair Programming with Respect to System Complexity and Programmer Expertise", *IEEE Transactions in Software Engineering*, 33(2), pp. 65-86, 2007.

[3] V. Basili, Shull, F.,Lanubile, F., "Building Knowledge through Families of Experiments", *IEEE Transactions on Software Engineering*, 25(4), pp. 456-473, 1999.

[4] K. Beck, *Extreme Programming Explained: Embrace Change*, Second ed. Reading, Mass.: Addison-Wesley, 2005.

[5] K. Beck, *Extreme Programming Explained: Embrace Change*. Reading, Mass.: Addison-Wesley, 2000.

[6] J. Carver, Jaccheri, L., Morasca, S., and Shull, F., "Issues Using Students in Empirical Studies in Software Engineering Education ", Proceedings of IEEE Metrics, pp. 239-249, 2003.

[7] A. Cockburn, *Agile Software Development*. Reading, Massachusetts: Addison Wesley Longman, 2001.

[8] A. Cockburn, *Crystal Clear: A Human-Powered Methodology for Small Teams*: Addison Wesley, 2004.

[9] T. D. LaToza, Venolia, G., DeLine, R., "Maintaining mental models: a study of developer work habits", Proceedings of International Conference on Software Engineering, pp. 492-501, 2006.

[10] L. Layman, L. Williams, and L. Cunningham, "Motivations and Measurements in an Agile Case Study", Proceedings of ACM SIGSOFT Foundation in Software Engineering Workshop Quantitative Techniques for Software Agile Processes (QTE-SWAP), Newport Beach, CA, 2004.

[11] F. Maurer and S. Martel, "Extreme Programming: Rapid Development for Web-Based Applications", *IEEE Internet Computing*, 6(1), pp. 86-91, Jan/Feb 2002.

[12] E. M. Maximilien and L. Williams, "Assessing Test-driven Development at IBM", Proceedings of International Conference of Software Engineering, Portland, OR, pp. 564-569, 2003.

[13] G. Melnik, Maurer, F., "A Cross-Program Investigation of Students' Perceptions of Agile Methods", Proceedings of International Conference on Software Engineering, pp. 481-488, 2005.

[14] K. Schwaber and M. Beedle, *Agile Software Development with SCRUM*: Prentice-Hall, 2002.

[15] H. Sharp, Robinson, H., "An Ethnographic Study of XP Practice", *Empirical Software Engineering*, 9(4), pp. 353-375, 2004.

[16] M. Stephens, Rosenberg, D., *Extreme Programming Refactored: The Case Against XP*: Apress, 2003.

[17] L. Williams, W. Krebs, L. Layman, A. Antón, and P. Abrahamsson, "Toward a Framework for Evaluating Extreme Programming", Proceedings of Empirical Assessment in Software Eng. (EASE) 2004, Edinburgh, Scot., pp. 11-20, 2004.