Association for Information Systems
AIS Electronic Library (AISeL)

SAIS 2008 Proceedings | Southern (SAIS)

3-1-2008

What Makes Agile Development Different?: A Case Study of Agile In Practice

Lewis Chasalow
Virginia Commonwealth University, chasalowlc@vcu.edu

WHAT MAKES AGILE DEVELOPMENT DIFFERENT?: A CASE STUDY OF AGILE IN PRACTICE.

Lewis Chasalow
Virginia Commonwealth University
chasalowlc@vcu.edu

ABSTRACT

Agile development methods have been described by proponents as being the best way to deal with the dynamic nature of software development in organizations, yet looking at agile practices of the major agile methodologies reveals many practices that have been used in the past. This work examined agile practices from the perspective of software development professionals and identified four characteristics of agile approaches that contribute to its perceived utility.

Keywords

Software development methodologies, agile development, scrum, extreme programming, case study.

INTRODUCTION

In 2001 some software development experts felt that processes and methodologies were becoming burdensome and getting in the way of successful implementation. They met to discuss what they could do to improve the situation and developed what they called *the agile manifesto* [1]. It states:

> **Manifesto for Agile Development**
>
> *We are uncovering better ways of developing software by doing it and helping others to do it. Through this work we have come to value*
>
> - Individuals and interactions over processes and tools
> - Working software over comprehensive documentation
> - Customer collaboration over contract negotiation
> - Responding to change over following a plan
>
> *That is, while there is value in the items on the right, we value the items on the left more*. [2]

Some of the methods that are now considered to be agile actually pre-dated this manifesto. Among the more common agile methods are *extreme programming* (XP) [3], *lean development* [4], and *scrum* [5]. Each of these follows the basic principles of the agile manifesto, but with different practices.

Examining the principles and practices of these three methods reveals that their primary focus is on the management of the software development team, not on the processes that deliver solutions, consistent with the first bullet of the agile manifesto. Yet most of the publications on agile development either focus on the benefit of one of the practices, such as *pair programming* [6-8], or provide prescriptions as to how to use one of the agile techniques [2, 9].

Abrahamsson, Warsta, Siponen, and Ronkainen observed with regard to the benefits of agile techniques that "scientific research is yet scarce. …and the empirical evidence is still very limited" [10]. Conboy and Fitzgerald observed "… that there is no consensus as to what constitutes an agile method." They further go on to say "that there is a fundamental problem with automatically declaring agile methods to be superior. There is a strong argument for assessing an SD [software development] team's need to be agile, and comparing that need to the capabilities that team has for being agile. In simple terms, if you don't need to be agile, or if you can't be agile, then don't be agile" [11].

Many of the concepts embodied in agile methods look very much like prototyping practices that have been around for more than 20 years [12]. Beck, one of the developers of extreme programming, observed that "the individual practices in XP are not by any means new. Many people have come to similar conclusions about the best way to deliver software in environments where requirements change violently" [13]. Agile methods rely on delivering working code in short cycles, having the working code reviewed by customers, and then following up with another iteration either improving on the capabilities just delivered, or adding additional features. The main difference espoused by texts on agile methods is the focus on very specific, prescriptive approaches to managing development teams.

Many other software development methodologies have been used since software was first written. The *waterfall* model assumes that software development is sequential, occurring in distinct stages. It uses the basic steps of requirements, design, development, testing, and implementation. This model is widely criticized for being too rigid, and unable to adjust to rapidly changing business requirements. In spite of this it is still the most prevalent process in use by practitioners [14].

Iterative approaches address some of the shortcomings of the waterfall model. In iterative development, rather than completing all of the development steps for a particular project at once, the project is completed through iterations during which some part of the end solution is delivered. Typically, each iteration delivers a working prototype. The prototypes evolve based on user input until the final solution is attained [15].

Agile techniques all follow some type of iterative prototyping approach, but are agile methods any different from older prototyping approaches? If agile is different, does it provide better results than either the waterfall model or other iterative approaches? The objective of this research is to determine if agile methods are more effective than conventional methods, and if so, to identify some of those characteristics that make them more effective. This paper examines the benefits of agile development from the perspective of practitioners participating in an agile project.

APPROACH

Benbasat, Goldstein, and Mead [16] observe that "IS researchers … often find themselves trailing behind practitioners in proposing changes or in evaluating the innovations put in place by practitioners," and that one of the important tasks of information systems (IS) researchers is "capturing the knowledge of practitioners and developing theories from it." They further suggest that case study research is well suited to this type of investigation. As agile methods were conceived and documented by practitioners, and this study seeks to gain knowledge regarding agile development techniques to allow development of theory, a case study approach seems appropriate.

Yin [17] defines a case study as "an empirical inquiry that investigates a contemporary phenomenon within its real-life context, especially when the boundaries between phenomenon and context are not clearly defined." Yin also contends that case study research is best suited to answering "how" or "why" questions. In this paper we investigate **how** agile methods are different and **why** they are effective.

The unit of analysis for this study is a single agile development team. Data was collected via direct observation of team interactions, and with semi-structured interviews of members of the team. The target organization is the information technology (IT) department of a large financial services company. The management of this department has been looking for ways to reduce the turnaround time for delivering system enhancements to the business. The company has over 400 IT development projects of various sizes underway at any point in time. Some of these projects take as little as a month, but most are much longer. The company has turned to agile methods with the goals of reducing the time to market for certain critical projects, and improving business customer satisfaction, by improving alignment of delivered solutions with true business needs.

A consulting firm with experience in agile methods has been hired by the IT department to support multiple pilot projects with an agile approach. The projects selected for the pilot were of medium size and with relatively near term deliverable dates. Members of the teams chosen to participate in the agile trials were software developers with experience working in the company's more traditional software development methodology, and some amount of outside experience. No new or inexperienced personnel were selected for the agile teams. Each agile team consists of an agile coach (from the consulting firm), a technical team leader, at least one software developer, at least one designer, a project manager, and a customer representative.

In this initial study one of these agile teams is examined. This team consists of a customer representative, a project manager/agile coach, a software designer, a developer, a database administrator, and a tester. Other resources are available to the team as needed. The team was assigned to an agile work room, which consisted of a large open room, with whiteboards on all of the walls, and large screen, networked workstations throughout the room.

OBSERVATIONS

The agile team under study used a modified version of the SCRUM agile method [5]. The modifications included the use of "user stories," an extreme programming technique, for documenting requirements [18], and the use of an internal company developed design template for capturing detailed software designs. Otherwise the team followed the seven prescribed SCRUM practices of:

1. ***Product Backlog*** – "…an evolving, prioritized queue of business and technical functionality that needs to be developed into a system."
2. ***Committed Teams*** – cross functional teams consisting of between 5 and 9 members dedicated to supporting the scrum effort.
3. ***Daily "Scrum" Meetings*** – a review by each team member of what they accomplished yesterday, what their plans are for today, and any issues impacting their ability to deliver.
4. ***Sprints*** – a 30 day period of development effort for a scrum team.
5. ***Incremental Product Delivery*** – solutions are delivered through a series of sprints, each one designed based on delivering the top priorities from the backlog.
6. ***Team Co-location***
7. ***Sprint Reviews*** – "… a four hour informational meeting. During this meeting the team presents to management, customers, users, and the Product Owner the product increment that it has built during the sprint". [5]

One of the main objectives of this study is to determine if agile methods produce different results from other iterative approaches or the traditional waterfall model. The observations were therefore described in the context of each of the scrum practices mentioned above to allow comparison of the agile approach to other approaches.

Product Backlog – The product backlog is a list of business and/or technical capabilities to be delivered by the project team. The concept behind a product backlog is that the team uses this to draw from when determining the content of each sprint. The product backlog was developed by the team based on the overall objective for the project and with participation of the team's business representative.

This practice relates to requirements elicitation and documentation. The objective of the product backlog is to describe all of the desired features, just as one would when gathering business requirements. Scrum doesn't specify a specific approach or tool for collecting or documenting the backlog; only that it should be created and maintained. This team used the XP practice of user stories to document the requirements that embodied the backlog.

The main difference between the product backlog approach and waterfall requirements gathering is that the backlog is built as the project progresses, rather than elicited up front. Some iterative approaches also do this, but others rely on up-front requirements gathering, with later design/development iterations. There was consensus on the agile team that the agile approach resulted in less rework, and better alignment of the final product with business needs. This was because the product backlog was built as the project proceeded, a business customer was involved at every step of the process, and the backlog could be re-prioritized throughout the project to make sure that the most critical features were worked on first.

A critical issue raised by the team was the need to properly document the details of each backlog item. Even though the agile manifesto states that they value, "working software over comprehensive documentation" [1], there is still a need for documentation to provide information for both application support and any subsequent enhancements. The systems analyst suggested that due to the quick turnaround time on delivering code, there often wasn't time to document what they were doing as they went along. He was concerned that it would end up not getting done, making subsequent work more difficult.

Committed Teams – Having a team consisting of diverse people with all of the skills required to complete the project was consistently cited as one of the reasons that they thought agile was working well. In the company's standard software development methodology, and in most traditional approaches, development resources are assigned to a project as needed.

Significant time may be wasted waiting for the right resources to become available to work on a particular project. By having a dedicated development team, with resources representing all of the skill sets required to deliver the solution, waiting time is largely eliminated.

One concern raised was idle time for certain people. The database administrator (DBA) in particular stated that there were times of the day or days of the week when he had very little to do. The nature of software development work is such that not every type of task is performed equally during a project lifecycle. Agile techniques may result in lower individual productivity. The team members stated that the total effort to deliver a solution was still lower than using traditional methods because idle time existed in other methods as well. More study is warranted in this regard.

Daily Scrum – The daily "stand-up," as they call it, occurred each morning in the project room. Each team member took 5 minutes to review what they had accomplished the day before, what their tasks were for the day, and raised any issue or pointed out any obstacles that they saw. Team members took a little time to get used to being totally open with each other when the team was first formed. After the first week or so they all seemed to be more comfortable with this process.

There is no real analogous process in traditional iterative or waterfall approaches. The type of information shared in these sessions is typically only shared with the development manager and project manager.

Overall the team members found this to be an effective way for everyone on the team to get a comprehensive understanding of project status. This process also provided an additional forum for team members to address issues before they became problems and to make sure that efforts weren't duplicated. Team members viewed this as an improvement over their experiences with traditional approaches. One problem though with this process was that the amount of information presented could be hard to handle.

Sprints – The team believed that working towards 30 day cycles kept them focused. It was a challenge for some to break the work down into units small enough to be accomplished in this timeframe. One person expressed a concern that there were items that required long lead-times and thus needed to be initiated in early sprints, but that they weren't always known until later sprints. So far this has not turned out to be a problem, but it was a concern of at least one person.

In most waterfall approaches it would be unusual for a particular step to take 30 days. Because of the long lead-time between requirements gathering and design and coding, the likelihood of requirements changes occurring before the end product is delivered is high.

Incremental Product Delivery – This is a prototyping concept that has been around for more than 20 years [12]. The team members all believed that the iterative approach worked for the scope of the project on which they were working, but raised concerns as to whether it would work for very large projects. Planning the iterations in useful sequences was raised as a concern. This contrasts directly with the structured steps of a waterfall methodology.

Team Co-location – Having all members of the agile team in the same room provided a number of benefits. They felt that it allowed the team to arrive at faster solutions to any issues. It helped all members of the team fully understand both the business and technical aspects of the project. In particular the business representative stated that she has a better understanding of the development process by being located with the developers, and the developers stated that they had a better appreciation of the total solution being provided. Co-location was only one contributing factor to this improved knowledge, but the team members believed that it was a critical factor.

Traditional waterfall or iterative methodologies generally don't address location of the development team. Team members can work from wherever they happen to be.

Sprint Reviews – Reviews were held with the business at the end of each 30 day sprint. This project consisted of a total of three sprints. Management was concerned after the first two that they didn't have a clear view as to whether the project was on track to complete the basic required capabilities. They were used to project status reports that gave them a percentage complete of a particular stage in the standard waterfall based development methodology in use in this company. After the third review they began to realize the progress that was being made and could see actual capabilities delivered. They observed that with the traditional approach they could see stated progress towards a milestone, but didn't really know if all capabilities would be delivered until the final stage when code was implemented.

Overall Observations – A consistent message from the team members was that the tasks that they perform in an agile project weren't really any different from those performed in a waterfall project. Four important features emerged as to what the team members saw as the main advantages of agile development over waterfall or even other approaches to prototyping, specifically:

1. Co-location of the team facilitates communication and collaboration.

2. Continuous access to a business representative facilitates continuous adjustment to requirements that assures alignment of results with objectives.

3. Short development cycles allow for faster reaction to change.

4. Empowerment of the team and the business representative allows for quicker decision making.

CONCLUSIONS

Based on the observations described above, we conclude that agile development does offer improvements over other methodologies. The software delivered by this team was used by the company for several months after the conclusion of the project and proved to be reliable and meet all of the businesses needs. They discontinued using the software when the product trial for which it was built was decided to be terminated, but the software was not a factor in that decision. The business observed that this software was delivered in approximately half the time and with an equivalent level of reliability as similar projects in the past using traditional techniques. The individual features that characterize agile methods are not very different or new, but it is the combined application of these features that seems to lead to more successful development. Based on our results, we conclude that agile development is more effective than waterfall or even other prototyping techniques in rapidly delivering solutions that closely meet business needs.

The features that make agile more effective are the co-location of the team, the access to a business representative, the short development cycles, and the increased empowerment, which together facilitate collaboration, rapid decision making, and continuous adjustment to change. In addition, morale seemed to be higher among the agile team members than what they had experienced in the past. This could be due to the extra attention they received by being part of a pilot project, rather than due to the agile approach itself. Further study is warranted to examine this question.

This study looked at a single agile team in a single company. The benefits identified were ascribed to practices that are not unique to agile methods. Collaboration, co-location, business/user involvement, and iterative development are not unique to agile approaches. Each of these has been studied independently. The question is still whether using all of these features together is required for agile methods to work. Additional study should be performed to see if there is a particular combination of processes that works better than other combinations.

REFERENCES

1. Highsmith, J. *History: The Agile Manifesto*. http://www.agilemanifesto.org/history.html 2001 [cited; Available from: http://www.agilemanifesto.org/history.html.
2. Martin, R.C., J.W. Newkirk, and R.S. Koss, *Agile Software Development: Principles, Patterns, and Practices*. 2003, Upper Saddle River, NJ: Pearson Education, Inc.
3. Beck, K. and C. Andres, *Extreme Programming Explained: Embrace Change*. Second ed. The XP Series, ed. K. Beck. 2005, Upper Saddle River, NJ: Addison-Wesley/Pearson Education.
4. Poppendieck, M. and T. Poppendieck, *Lean Software Development: An Agile Toolkit*. The Agile Sfotware Development Series, ed. A. Cockburn and J. Highsmith. 2003, Upper Saddle River, NJ: Addison Wesley/Pearson Education.
5. Schwaber, K. and M. Beedle, *Agile Software Development with Scrum*. 2002, Upper Saddle River, NJ: Prentice Hall.
6. Parrish, A., et al., *A Field Study of Developer Pairs: Productivity Impacts and Implications.* IEEE Software, 2004. **21**(5): p. 76-79.
7. Nosek, J.T., *The Case for Collaborative Programming.* Communications of the ACM, 1998. **41**(3): p. 105-108.
8. Williams, L., et al., *Strengthening the Case for Pair Programming.* IEEE Software, 2000. **17**(4): p. 19-25.

9. Succi, G. and M. Marchesi, eds. *Extreme Programming Examined*. The XP Series, ed. K. Beck. 2001, Addison Wesley: Upper Saddle River, NJ.
10. Abrahamsson, P., et al. *New Directions on Agile Methods: A Comparative Analysis*. in *Proceedings of the 25th International Conference on Software Engineering* 2003: IEEE Computer Society
11. Conboy, K. and B. Fitzgerald. *Toward a Conceptual Framework of Agile Methods*. in *4th Conference on Extreme Programming and Agile Methods*. 2004. Calgary, Cn: Springer.
12. Naumann, J.D. and A.M. Jenkins, *Prototyping: The New Paradigm for Systems Development.* MIS Quarterly, 1982. **6**(3): p. 29-44.
13. Beck, K., *Embracing Change with Extreme Programming.* IEEE Computer, 1999. **32**(10): p. 70-77.
14. Laplante, P.A. and C.J. Neill, *"The Demise of the Waterfall Model Is Imminent" and Other Urban Myths.* Queue, 2004. **1**(10): p. 10-15.
15. Goldman, N. and K. Narayanaswamy. *Software evolution through iterative prototyping*. in *The 14th International Conference on Software Engineering*. 1992. Melbourne, Australia: ACM Press.
16. Benbasat, I., D. Goldstein, and M. Mead, *The Case Research Strategy in Studies of Information Systems.* MIS Quarterly, 1987. **11**(3): p. 369-386.
17. Yin, R.K., *Case Study Research Design and Methods*. Applied Social Research Methods Series. Vol. 5. 2003, Thousand Oaks, CA: Sage Publications, Inc.
18. Cohn, M., *User Stories Applied: For Agile Software Development* 2004, Addison-Wesley Professional