

Implementation of Agile Methods for Large Projects and Critical Requirements

Avishek Shrestha (avishek@err7.net)

Although opponents of agile methods are quick to point out that agile methods are thought to be inappropriate for large projects or projects with critical requirements, properly implemented agile methods are in fact suitable for those very same projects that the opponents cite as examples. Furthermore, with proper implementation, agile methods are more efficient with fewer resources than traditional development methods. This paper identifies several recommendations to help agile teams develop large projects and identify critical requirements through the emphasis on documentation, practice of understanding critical requirements, using agile iterative development to address changes in requirements and assess milestones, and utilizing small, flexible teams with frequent communication.

About Agile Methods

In order to understand the efficiency and capability of the agile methods, it is first important to understand the doctrines of the agile methods manifesto. Agile methods are efficient because they allow an organization to do more with less. Simply put, small teams using iterative processes and frequent communication are able to create quality products that rival larger organizations with larger teams and defined non-iterative prescriptive processes.

Agile methods follow the rules of the agile manifesto [2]:

- Customer satisfaction by rapid, continuous delivery of useful software
- Working software is delivered frequently (weeks rather than months)
- Working software is the principal measure of progress
- Even late changes in requirements are welcomed
- Close, daily cooperation between business people and developers
- Face-to-face conversation is the best form of communication
- Projects are built around motivated individuals, who should be trusted
- Continuous attention to technical excellence and good design
- Simplicity
- Self-organizing teams
- Regular adaptation to changing circumstances

The four values of the agile manifesto include [4]:

- We value individuals and interactions over processes and tools.
- We value working software over documentation.

- We value customer collaboration over contract negotiation.
- We value responding to change over following a plan.

The Benefits of Agile Methods

The main benefit of the agile methods is that it allows for an adaptive process - in which the team and development react to and handle changes in requirements and specifications, even late in the development process. Through the use of multiple working iterations, the implementation of agile methods allows the creation of quality, functional software with small teams and limited resources.

Proponents of the traditional development methods criticize the agile methods for the lightweight documentation and inability to cooperate within the traditional work flow. However, with proper implementation the agile methods can complement and benefit traditional development methods. I would suggest that proponents of traditional development methods evaluate the doctrines of agile methods in order to better understand how agile methods can complement traditional development methods. Furthermore, it should be noted that traditional development methods in non-iterative fashions are susceptible to late stage design breakage. Agile methods effectively address this by frequent incremental builds which encourage changing requirements. The following recommendations, described in detail, suggest implementations of the agile method for large project and projects with critical requirements

Recommendation 1: An Agile Focus On Documentation

Firstly, it is important to acknowledge that the agile team must recognize the significance of documentation. Documentation is emphasized in traditional development methods because the care requirements are well defined. However, traditional development methods fail to accommodate changes in requirements like agile methods do. One misperception of agile methods is that they hold little or no value in documentation and key plans [1]. Agile teams that shun and neglect documentation must evaluate the significance of documentation in relation to the expectations of management, clients, end-users and other stake-holders. However, this diligence to documentation need not come at the expense of a working product, and the agile team can still prioritize working products over documentation - as long as documentation is not neglected. Accordingly, I propose that the agile team lead assign two members to produce documentation in parallel and concurrence with development. The two members will be responsible for writing, reviewing, and maintaining documentation consistent with development. Furthermore, efficient practices like peer reviews will help to ensure the accuracy and quality of the documentation. Although this may detract resources from development, this will help to ensure that the agile team recognizes the significance of documentation in relation to the project in its entirety, without significantly delaying the end product.

Recommendation 2: Understanding Critical Requirements

Secondly, the agile team needs to understand the importance of critical requirements. However, this responsibility lies with stake-holders rather than just the agile team. If reliability is a critical requirement, this must be conveyed to the agile team. If security is a critical requirement, this must be conveyed to the agile team. The significance of critical requirements needs to be conveyed to the agile team prior to the commencement of the project. In order to accomplish this, I propose that the customer and agile team leaders arrange for meeting to discuss critical requirements in the earliest stages. The customer and agile team leaders are responsible for identifying critical requirements. Because the agile team leaders themselves are responsible for identifying possible critical requirements, agile team leaders will have to be well experienced with expertise in areas regarding critical requirements. Large projects are projects which meet the following characteristics [3]:

- 200+ people
- Distributed across multiple time zones
- Workers report to three different companies
- The lifecycle is 2.5 years to first delivery and 6 months for subsequent updates

One of the most important aspects of determining critical requirements in an agile team is distinguishing “must have” requirements from “nice to have” requirements. The importance of this cannot be emphasized enough, and the agile team must make deliberate efforts into distinguishing what the “must have” requirements are. Frequent communication with the customer, especially in the early stages of the development. Furthermore, frequent communication with the customer enables the agile team to resolve any ambiguity in determining requirements of both sorts: the “must haves” and the “nice to haves” [5].

Once the initial critical requirements of a large project have been identified and documented, the agile team can begin with development. It is important to note that the agile team will not discourage changing requirements further along in the development cycle - but that changes to the requirements must wait until the culmination of each iteration. Consequently, this methodology will ensure that iterations are consistent with expectations, and the development process will remain organized. Opponents of the agile methods are eager to paint the agile methods as chaotic and unorganized, but this is far from the truth - as planning is actually a core principle of agile methods [1]. Agile methods will thrive with organization and structure, and the agile team must remain organized within iterations in order to deliver according to expectations.

Recommendation 3: Agile Advantages Of Iterative Development

Thirdly, iterative development of agile methods allows early assessment of design weaknesses. Proponents of traditional development methods imply that agile methods are not suitable to

handle large projects and that only traditional development methods are able to handle large projects. Traditional development methods like the waterfall model emphasize design (and the corresponding design documentation) prior to the commencement of development. However, this can be counter-productive as early design models can help to identify weaknesses within the design itself or to recognize more efficient and optimal solutions. Agile methods, which place an emphasis on working products, can help to overcome this weakness of the traditional development model by providing a product early in the development cycle. To do this, I propose that several agile teams work in parallel on different aspects of the larger project. Doing so will ensure that communication and interfacing problems will arise between development groups. To counter the communication and interfacing problems, the agile teams should arrange frequent conference calls and meetings to limit requirements creep and/or scope creep.

Agile teams employing modern communications methods like instant messaging, web based shared team projects, and discussion boards are best able to harness the power of modern communications methods previously unavailable for the rapid development techniques enjoyed by the agile methods. Furthermore, frequent communication between agile teams will encourage the teams to engage in constant cooperation. Although several agile teams working in parallel on different areas of the large project may introduce other problems, I believe those problems can be overcome with better communication. At the culmination of an iteration, I suggest that the agile teams have a demonstration of the working product as well as a post mortem analysis and discussion. It is important to determine what and act based on what matters to the customer and that the customer and developer fail to collaborate sufficiently in reaching common agreements [1]; a post mortem analysis and discussion would serve to ensure that the customer and agile development teams have a clear understanding of the current progress and future expectations of progress.

The decomposition of the larger parts of the project into smaller components, called sub-components, lends itself to the employment of more agile teams. These agile teams can even work in other time zones and other countries provided that these agile teams keep frequent communication and are self organizing. Proponents of traditional development methods like the waterfall model emphasize heavy-weight documentation prior to development, but these proponents fail to see that the decomposition of larger parts of the project fits perfectly within the agile methods doctrine. Agile teams working in parallel on sub-components allows for quick development and an early design. An early design leads to an early review. Consequently, the iterative schedule and emphasis on deliverable products allows the agile teams to assess the successes and shortcomings, and plan for the next iteration. Once a specific agile team has successfully completed a sub-component, the team is available to work on another component or sub-component. Each of these smaller agile teams will still be responsible for assigning two members to complete the previously described documentation which is necessary to satisfy the other stakeholders.

Recommendation 4: Smaller Agile Teams Are Flexible

Lastly, flexible agile teams are better able than lumbering traditional teams to address the changes in requirements and expectations that arise from iterative development schedules. Agile teams are able to react to changes that would hinder the traditional development methods with late stage breakage. After having addressed the critical requirements and developed an early design for review, the agile team is able to shift focus to noncritical requirements after approval of the critical requirements. Furthermore, I suggest that agile teams maintain frequent, if not daily, communication with other stakeholders. This discussion will direct the expected goals for the next iteration - whether the expected goals are to refine specific areas of the previous iteration or to begin development on new areas. Agile teams have the benefit of being able to quickly shift focus while working concurrently to deliver a quality end product faster than a team following a traditional development model.

One of the common misperception of agile methods is that the requirements changes are not sufficiently controlled [1]. Although requirements changes are encouraged in agile methods, the changes are also strictly controlled. Frequent communication encourages the customer to embrace the concept of requirements change - but the agile team must additionally employ methods that control the changes. So that although changes are encouraged, they must be controlled in order to continue with progress with minimal setbacks. Smaller agile teams utilizing frequent communication are still able to encourage change while also controlling those changes.

Conclusion

In conclusion, although the traditional development methods have been used for large projects and projects with critical requirements, careful implementation of the agile methods emphasize that agile methods are also suitable for the same projects as outlined in the above recommendations. Provided that the agile teams recognize the significance of documentation and identification of critical requirements, agile methods are suitable for handling projects with critical requirements. Additionally, agile teams working in parallel with frequent communication between teams are well equipped to handle large projects. At the culmination of each iteration, the agile teams will provide a demonstration, post mortem analysis and discussion, and review of documentation. Consequently, properly implemented agile methods are viable and competent solutions to the very projects that proponents of traditional development methods are quick point out are not suitable for agile methods.

Works Cited:

[1] McMahon, Paul F. "Bridging Agile and Traditional Development Methods: A Project Management Perspective." Reifer, Donal J. Software Management. Hoboken, NJ: IEEE Computer Society, 2006. 49-53.

[2] Wikipedia. Agile Software Development. March 2008
<http://en.wikipedia.org/wiki/Agile_software_development>.

[3] IBM. Scaling Down Large Projects To Meet The Agile Sweet Spot. April 2008.
<<http://www.ibm.com/developerworks/rational/library/content/RationalEdge/aug04/5558.html>>.

[4] Cockburn, Alistair. Agile Software Development. Addison-Wesley, 2002: 215-218.

[5] CTSC Crosstalk. Lessons Learned Using Agile Methods on Large Defense Contracts . April 2008. <<http://www.stsc.hill.af.mil/CrossTalk/2006/05/0605McMahon.html>>