Bridging Agile and Traditional Development Methods: A Project Management Perspective

Paul E. McMahon
PEM Systems

Monday, 19 April 2004
Track 4: 3:35 - 4:20
Ballroom D

Today, companies are reporting success in meeting rapidly changing customer needs through agile development methods. Many of these same companies are finding they must collaborate with organizations employing more traditional development processes, especially on large Department of Defense projects. While it has been argued that agile methods are compatible with traditional disciplined processes, actual project experience indicates conflicts can arise. This article identifies specific project management conflicts that companies face based on actual project experience, along with strategies employed to resolve these conflicts and reduce related risks. Rationale, insights, and related published references are provided along with lessons learned and recommendations. If you work for a company that is using or considering using agile development, or your company is collaborating with a company using an agile method, this article will help you understand the risks, issues, and strategies that can help your project and organization succeed.

This article was motivated by a case study where a small company using a well-known agile method – eXtreme Programming (XP) – requested help addressing specific conflicts that arose on the project where they were a subcontractor to a larger organization employing a traditional development method. The purpose of this article is not to compare agile and traditional methods, but to raise awareness of potential project management conflicts that can arise when a company employing an agile method collaborates with a company employing a traditional development methodology. It also identifies practical steps that can be taken to reduce related risks.

It is worth noting that the case study presented is not unique. Published references documenting similar conflicts are provided. Also notable is that the motivation for examining this project extends beyond the case study itself. Today there exist increasing opportunities for small companies to gain new work through software outsourcing from traditional development organizations.

Where Are We Going?

In this article, I first identify key case study facts along with relevant information and common misperceptions related to traditional and agile methods. Next, I identify four conflicts observed along with five recommendations and one lesson learned. The company named SubComp refers to the subcontractor employing an agile methodology. The company named PrimeComp refers to the prime contractor employing a traditional development methodology.

Case Study Key Facts

Shortly before I was asked to help SubComp, PrimeComp's customer had withheld a progress payment based on a perceived risk observed at a recent critical design review (CDR). Written comments provided to PrimeComp indicated that the customer wanted to see working software in order to *assess the proposed design and related risk*. The area of concern was SubComp's responsibility. Upon receiving the customer comments, PrimeComp requested that SubComp provide additional detailed design documentation.

It is important to note that PrimeComp required all correspondence between the customer and SubComp to go through them. It is also important to note that most of the contractually required documentation was not formally due until the end of the project, and, prior to the CDR, little had been communicated to SubComp with respect to documentation content and expectations. The project was planned using a traditional waterfall life cycle with a single CDR.

Early in the project, SubComp had identified multiple technical risks. However, it had decided in the early stages to focus its small agile team on a single technical risk that it had assessed to be of much greater significance than all other risks. At the recent CDR, SubComp had provided a demonstration with working software that addressed this risk to the customer's satisfaction.

The risk the customer was currently raising was one SubComp viewed as lower priority. To address this risk, the XP team was focusing on a second demonstration with working software to show to the customer at a follow-up CDR. In parallel, it was also driving to meet PrimeComp's request for additional detailed design documentation. This follow-up CDR had not originally been planned, and it was causing project tension because of the progress payment holdup.

Agile Development Methods

In the spring of 2001, 17 advocates of agile development methods gathered in Utah and agreed to a set of four values and 12 principles referred to as the Agile Manifesto [1]. The four values are expressed in the following:

> We are uncovering better ways of developing software by doing it and helping others 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. [1]

One misperception of agile methods is that they hold little or no value in documentation and plans. Note that the value statements express a relative value of documentation and plans with respect to working software and responding to change.

Traditional Development Methods

The traditional waterfall model is well known, but it is important to understand that it is not an incremental model. Today, Department of Defense policy 5000.1 and 5000.2 [2] strongly encourages using the spiral model for software development. Although the spiral model was first introduced by Barry Boehm in the mid-80s [3], the risk-driven essentials of the model frequently have been misunderstood. To help clarify, Boehm recently identified six spiral essentials [2]:

- Concurrent determination of key artifacts.
- Stakeholder review and commitment.
- Level of effort driven by risk.
- Degree of detail driven by risk.
- Use of anchor-point milestones.
- Emphasis on system and life-cycle artifacts (cost/performance goals, adaptability).

A Perspective on Waterfall, Spiral, and Agile Development

Historically, the waterfall life-cycle model has been closely associated with heavyweight documentation. The spiral model has also historically been misinterpreted as an incremental waterfall model, rather than as a risk-based model as clarified by Boehm. It is important to note the focus on people (individuals, stakeholders), products (working software, key artifacts), and change (responding to change, adaptability) common to both the Agile Manifesto and the spiral essentials.

Methods Compatibility or Conflict?

It would seem from this observation that a company using an agile methodology should be able to successfully collaborate with a company using a traditional development method, especially if the project contained risk. To further support this position, Mark Paulk, co-author of the Software Engineering Institute's Capability Maturity Model® (CMM®), which has been associated with rigorous traditional development methods, has stated, "XP is a disciplined process, and the XP process is clearly well defined. We can thus consider CMM and XP complementary" [4].

Despite this evidence of methods compatibility, a different situation appears to exist in the developmental trenches. This is clearly pointed out by Don Reifer in the following statement made in reference to one of his own studies:

> Instead of trying to make XP work rationally with the firm's existing processes, each side is pointing fingers at the other. No one seems to be trying to apply XP within the SW-CMM context rationally and profitably as the sages have suggested ... XP adherents feel they don't have time for the formality … Process proponents argue … quality will suffer and customer satisfaction will be sacrificed. [5]

One proposal to address this conflict has been put forth by Scott Ambler in the form of a *blocker* who runs *interference* for the team by providing, in Ambler's words, "the documents that the bureaucrats request" [6]. The term blocker essentially means someone whose sole job is to keep non-agile project stakeholders from hindering the agile development team's progress.

"Using an incremental life-cycle model is critical because many of the conflicts observed are rooted in the all up-front thinking that comes with the single-increment waterfall model. Incremental thinking is fundamental to agile methods and crucial to bridging the two methods."

In the following paragraphs we identify four conflicts observed on the PrimeComp case study project.

Conflict 1: Working Software vs. Early Design Documentation

Part of the difficulty faced by SubComp is the conflict between what they perceive the end-customer wants with respect to risk management, and what is being asked for by their immediate-customer, PrimeComp. The end-customer has asked to *see working software*. It appears that the end-customer wants more than a *paper* design to assess the risk, yet PrimeComp is asking SubComp to provide more detailed design documentation.

Conflict 2: Single vs. Multiple-Increment Life Cycle

Assuming SubComp succeeds in addressing the immediate high visibility risk, what if yet another risk pops up at the follow-up CDR? Will there be a follow-up to the follow-up CDR with further delays of progress payments?

Agile teams often tackle tough issues first, as did SubComp. They focus on achieving customer satisfaction through frequent software deliveries based on clear priorities. Agile teams are also usually small and often do not have adequate resources to work multiple risks in parallel. The single iteration through the waterfall model with the planned single CDR milestone was a major project management conflict for the agile team. From a project management perspective, it was a critical conflict since a significant payment was withheld.

Conflict 3: Formal Deliverable Documentation Weight

Hearing that the documentation was not due until the end of the project led me to ask two questions:

- What exactly were the contractual documentation requirements?
- Did SubComp know PrimeComp's documentation expectations?

Companies that employ agile methods tend to provide *lightweight* documentation. This is, at least partially, because they value working software more than documentation. Although large documents are not a requirement of traditional development methods, cultural expectations often tend to the *heavyweight* side.

Waiting to deliver documentation until late in the project creates a potential conflict, especially if expectations have not been set through early communication. Because of the stress being placed on the agile team to complete the demo software and to provide additional detailed documentation, the possibility of using a blocker was considered.

Conflict 4: On-Site Customer Collaboration

Agile methods *embrace* [7] changing requirements, even late in development. During my discussions with SubComp personnel, I discovered that the contractual project requirements were, in the words of one team member, "high level and

® CMM is registered in the U.S. Patent and Trademark Office by Carnegie Mellon University.

open to many interpretations." When contractor and customer work closely on an agile development project, embracing change is eased through close collaboration. When a prime contractor inserts itself in the middle, effective collaboration can be stifled, creating additional conflict and risk.

During my discussions with SubComp personnel, one member of the agile team said,

> What is difficult from my perspective is that the details they [PrimeComp] are asking for are the lowest risk and lowest value technically … spending time on what they feel they need puts our small team at risk for actually completing the product, which is what ultimately matters, I think.

This statement led me to stop and consider just what ultimately did matter. The agile team was addressing the technical risks in a planned manner, but a key ingredient to effective XP operation was missing. How could the agile team determine and act based on what ultimately mattered to the customer when the customer was not collaborating closely on-site?

Given these four observed conflicts, what strategies make sense and what can be done to bridge agile and traditional development?

Recommendation 1: Plan Collaboratively and Use an Incremental Life Cycle

Some believe those who use agile methods do not follow a plan. This is a misunderstanding. Planning is actually a core principle of agile methods [7]; however, agile teams tend to plan in smaller time increments and more frequently than those who use traditional methods. This distinction has been clarified by the characterization of XP as *planning driven* rather than *plan driven* [8]. Planning takes place with both methods, but with agile methods the focus is on the act of planning, rather than a document.

I recommend for similar projects that the initial planning be done collaboratively, with the prime contractor, subcontractor, and customer working closely. I also recommend that an incremental life-cycle model be employed to aid in aligning the agile subcontractor's work within the overall project schedule. Using an incremental life-cycle model is critical because many of the conflicts observed are rooted in the all up-front thinking that comes with the single-increment waterfall model. Incremental thinking is fundamental to agile methods and crucial to bridging the two methods.

In the case study, it was too late to replan the project with an incremental life cycle, but I did recommend that SubComp step back and re-evaluate their strategy to the upcoming follow-up CDR. The following questions needed to be answered:

- Were there other risks that the customer felt had to be addressed at this point for the project to move forward successfully?
- What else would it take to close the CDR?

We had to find out the customer's CDR completion criteria, but we had to do it within an agile mind-set. That is, quickly and with minimal resources given that the small agile team was already overextended working to complete the demo and the prime contractor's request for additional documentation.

"The frequent feedback from multiple spirals can help your risk mitigation visibility and ultimately your project's success."

Recommendation 2: Use the Spiral Model and Well-Defined Anchor-Point Milestones to Address Risk

If one of the risk-responsible collaborating team members is employing an agile method, a spiral model with well-defined anchor points [2, 9] can go a long way to reduce potential project conflict and risk. This model can help the traditional development prime contractor as well as the agile subcontractor.

Providing working software early to address high-risk areas makes sense, but it is not sufficient to meet all project management needs. If you are the prime contractor, you want to make sure all the risks are being addressed in a timely and prioritized fashion. The frequent feedback from multiple spirals can help your risk mitigation visibility and ultimately your project's success.

From the agile subcontractor's perspective, you want your team to be able to focus and solve the highest priority risks early, but you also want to know what the prime contractor's expectations are along the way. By agreeing to the anchor-point milestones during the early collaborative planning activity, expectations can be made clear on both sides, allowing the project to operate more effectively. One of the reasons an incremental life-cycle model is recommended is because it often leads to earlier communication concerning priorities and risks. When a traditional waterfall life-cycle model is selected, early discussions concerning priorities are often missed.

Recommendation 3: Plan for Multiple Documentation Drops and Use a Bridge Person

One reason collaborative initial planning is recommended is to get discussions going early concerning product deliverables thereby reducing the likelihood of late surprises. In the case study, it was decided not to use a *blocker* to provide the contractual documentation. Discussions led the agile team to the conclusion that the blocker notion brought with it a negative view of documentation. The blocker was seen as someone outside the agile team whose job it was to develop the documentation without *bothering* the team. This was not consistent with SubComp's view of documentation, nor was it consistent with the values expressed in the Agile Manifesto.

Our solution was to use what we called a *bridge* person, rather than a blocker. The bridge person, unlike the blocker, was a member of the agile team who participated in team meetings providing a valuable service to the team by capturing key verbal points and whiteboard sketches thereby providing useful maintainable *lightweight* documentation that would ultimately help both contractors and the customer.

I recommended that multiple drops of documentation be provided to PrimeComp prior to the contractual delivery date to get early feedback and reduce late surprises. Waiting until late in the game to deliver documentation is risky, especially when expectations are uncertain.

Recommendation 4: Find a Way to Make Customer Collaboration Work for Effective Requirements Management

The reason establishing a close collaborative working relationship with the customer was not easy in our case study was because PrimeComp was sensitive to any contact between the end-customer and SubComp. This sensitivity was, at least partially, motivated by the desire to maintain control. It is also possible that uncertainty surrounding how the end-customer would perceive the use of an agile methodology fueled PrimeComp's desire to maintain a separation between

SubComp and the end-customer.

A recommendation I would make today to a prime contractor facing similar situations is to recognize that the key to maintaining real project control is the management of risks associated with the subcontractor's effort. One of the most common risks in similar situations is requirements creep, which often fails to get recognized until late in the project's test phase when the customer starts writing new discrepancies because the product does not meet what they now perceive the requirements to mean.

This situation frequently occurs because the end-customer and product developer (agile subcontractor) fail to collaborate sufficiently in the early stages reaching common agreements on potentially ambiguous requirements statements. In an agile development environment, this risk increases because requirements are often written as *story cards* [7], which have an implicit dependence on face-to-face communication to resolve potential differing requirements interpretations.

One common misperception of agile methods is that the requirements are not controlled since they are allowed to change late in the game. This is a legitimate concern that could also fuel why a prime contractor might want to keep an end-customer away from an agile subcontractor, but it also demonstrates a fundamental misunderstanding of agile methods.

As Craig Larman explains, "Iterative and agile methods embrace change, but not chaos" [10]. The following rule clarifies the distinction: "Once the requests for an iteration have been chosen and it is under way, no external stakeholders may change the work" [10].

If you are a prime contractor, I recommend that you check with your agile subcontractor to ensure they understand this crucial distinction between embracing change and living in chaos. I also recommend that a member of the prime contractor's team be placed on the agile team. Then encourage and support as much customer collaboration as you can between the agile team and the end-customer to help manage your own risk.

If you are an agile subcontractor, you want to demonstrate to the prime contractor that you are effectively managing your allocated requirements. Those employing agile methods often use *story point* [11] charts to depict work remaining and requirements creep. While story points and anchor points are different, they can be used together to help bridge the two methods.

Anchor points can be viewed as spiral model progress checkpoints [2, 9]. One weakness with traditional approaches has been the accuracy of the methods employed to measure progress. Story points provide an objective progress-measurement method based on stories verified through successfully completed tests [11].

At the start of each increment, I recommend that the agile subcontractor provide the prime contractor with a documented list of agreed-to stories to be completed in the upcoming increment. Story point charts can then be used to objectively back up anchor-point progress assessments, leading to improved team communication and trust.

Recommendation 5: Document and Communicate Your Process

I recommended to SubComp that they put together a presentation documenting their agile process from the project management perspective. This presentation would include key terms, roles, and responsibilities. Terms unique to the agile method such as coach, tracker, and metaphor [7] would be mapped to traditional terms such as project manager and architecture.

> *"If you are an agile subcontractor, you want to demonstrate to the prime contractor that you are effectively managing your allocated requirements."*

Recommendations for incremental lifecycle model, contract deliverables, and reviews compatible with both agile and traditional development methods should be included. Key to the presentation is a description of how SubComp's agile method fits within a traditional project management framework using a spiral model focusing on risk management. I then recommended to SubComp that they take every opportunity to communicate the key points in the presentation to PrimeComp, the end-customer, and to other traditional development contractors who might hold potential new business opportunities through software outsourcing.

Lesson Learned

When on-site customer collaboration exists, conflicts associated with vague requirements can often be resolved quickly. However, when the customer is not easily accessible, an agile subcontractor with vague requirements can quickly be placed at great risk.

We have learned that today's multi-contractor collaborative projects often do not lend themselves well to full-time, on-site customer collaboration. However, this does not mean that these projects cannot benefit from agility.

In such cases, I recommend a hybrid agile method with a focus on a more traditional requirements development and management method. Successful hybrid agile methods are not new [8, 12]. Keep in mind that hybrid does not imply all requirements up-front, but it does imply that once an iteration is under way, requirements must remain fixed to avoid chaos.

Conclusion

Today, we know how to manage geographically distributed teams formed from companies with divergent cultures and experiences [13]. Bridging agile and traditional development is the next step for organizations looking to take advantage of increasing new business opportunities through collaboration.

If you are experiencing conflicts similar to what has been described in this article, first examine your lines of communication. Look at your terminology. Are you communicating effectively what it is you do and how you do it? If you heard recently that a customer review did not go well, consider that the cause could be as simple as your agile terminology not connecting to the ear of a listener familiar only with traditional methods.

Allowing late requirements changes can work when your customer is on-site working next to you. But if you do not have an on-site collaborative customer, consider a hybrid approach to avoid major trouble late in the project.

Consider bridging, rather than blocking to meet milestone deliverables. More importantly, consider communicating to your collaborative partners and customers through examples of your products to gain their buy-in early, including the weight of your proposed documentation. Let them know that being agile is not cheating, but is in the best interests of everyone.

While many of the solutions described in this article are similar to those employed on non-agile projects, these solutions should not be taken for granted for two reasons. First, too often on hybrid agile-traditional projects, we find emotion

Coming Events

June 2-4
Symposium on Access Control Models and Technologies 2004
Yorktown Heights, NY
www.sacmat.org

June 11-13
ACM Sigplan 2004 Conference on Language Compilers and Tools for Embedded Systems
Washington, DC
http://lctes04.flux.utah.edu

June 14-17
SEPG Europe 2004
London, England
www.espi.org/frm–sepg.asp

June 14-18
ICSPI 2004 International Conference on Software Process Improvement
Washington, DC
www.icspi.com

June 23-26
Agile Development Conference 2004
Salt Lake City, UT
www.agiledevelopment conference.com

June 27- July 2
USENIX Annual Technical Conference
Boston, MA
www.usenix.org/events/usenix04

July 21-25
CITSA 2004 International Conference on Cybernetics and Information Technologies, Systems, and Applications
Orlando, FL
www.infocybernetics.org

April 18-21, 2005
2005 Systems and Software Technology Conference
Salt Lake City, UT
www.stc-online.org

getting in the way of clear thinking, often leading to a fundamental breakdown of communication. Second, we are learning through agile methods more effective techniques to measure progress and communicate. As Robert Martin pointed out in referring to agile methods:

> They're not a regression to the cave, nor are they anything terribly new: Plain and simple, the agile bottom line is the production of regular, reliable data – and that's a good thing. [11]

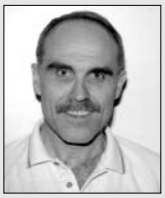
Don Reifer said, "No one seems to be trying to apply XP within the SW-CMM context rationally and profitably as the sages have suggested" [5]. In our case, studying the proactive steps taken based on the recommendations led to a successful follow-up CDR and to improved communication and early documentation agreements. Today, SubComp recognizes the value of XP, but they also recognize the value and need for fundamental project management, and they are looking to the CMM Integration℠ framework [14] to help guide related improvements.

Agility is not counter to effective project management, but agile methods do not provide all of the project management needs necessary for success. Wrap your agile development process in a lightweight project management framework, and watch your communication and collaboration improve and your project and company succeed.◆

References

1. Cockburn, Alistair. Agile Software Development. Addison-Wesley, 2002: 215-218.
2. Boehm, Barry. "Spiral Model as a Tool for Evolutionary Acquisition." CrossTalk May, 2001 <www.stsc.hill.af.mil/crosstalk/2001/05/index.html>.
3. Boehm, Barry. A Spiral Model of Software Development and Enhancement. Proc. of An International Workshop on Software Process and Software Environments, Trabuco Canyon, CA., Mar. 1985.
4. Paulk, Mark. "Extreme Programming from a CMM Perspective." IEEE Software Nov./Dec. 2001: 19-26.
5. Reifer, Don. "XP and the CMM." IEEE Software May/June 2003: 14-15.
6. Ambler, Scott. "Running Interference." Software Development July, 2003: 50-51.
7. Beck, Kent. Extreme Programming Explained: Embrace Change. Addison-Wesley, 2000.
8. Boehm, Barry, and Richard Turner. Balancing Agility and Discipline: A Guide for the Perplexed Addison-Wesley, 2003: 33-34, 233.
9. Boehm, Barry, and Daniel Port. "Balancing Discipline and Flexibility With the Spiral Model and MBASE." CrossTalk Dec. 2001 <www.stsc.hill.af.mil/crosstalk/2001/12/boehm.html>.
10. Larman, Craig. Agile and Iterative Development: A Manager's Guide. Addison-Wesley 2003: 14.
11. Martin, Robert C. "The Bottom Line." Software Development Dec. 2003: 42-44.
12. McMahon, Paul E. "Integrating Systems and Software Engineering: What Can Large Organizations Learn From Small Start-Ups?" CrossTalk Oct. 2002: 22-25 <www.stsc.hill.af.mil/crosstalk/2002/10/mcmahon. html>.
13. McMahon, Paul. E. Virtual Project Management: Software Solutions for Today and the Future. St. Lucie Press, 2001.
14. CMMI Product Team. Capability Maturity Model® Integration (CMMI®), Version 1.1. Pittsburgh, PA: Software Engineering Institute <www.sei.cmu.edu>.

About the Author



Paul E. McMahon, principal of PEM Systems, provides technical and management services to large and small engineering organizations. He has taught software engineering at Binghamton University; conducted workshops on engineering process and management; and published over 20 articles, including articles on agile development and distributed development in CrossTalk, and a book on collaborative development, "Virtual Project Management: Software Solutions for Today and the Future." McMahon also presented at the 2003 Software Technology Conference on "Growing Effective Technical Managers."

PEM Systems
118 Matthews ST
Binghamton, NY 13905
Phone: (607) 798-7740
E-mail: pemcmahon@acm.org