

How Can Collaboration Systems and Social Media Complement Agile Project Management?

Thursday, June 12, 2008 at 11:50AM

Dennis D. McDonald in Collaboration, Project Management, Social Networking, Social Media, Project Blog Survey, Agile

By **Dennis D. McDonald**

Introduction

I recently met with Thad Scheer, President of **Sphere of Influence, Inc.** Thad's company specializes in agile and lean project management methods.

During our meeting Thad and I discussed some of the changes occurring in project management methods and practices. Thad recommended I read an article by his colleague Erik Stein titled "Innovation: Agile with Intent." Stein's article was published in the October 2007 issues of the subscription-only **Cutter IT Journal** and starts like this:

In and of themselves, agile practices do not foster innovation. In fact, left untended in product development situations, they are quite likely to become obstacles to innovation.

In the article Stein explains how agile project management techniques may actually retard innovation in software development projects. I was struck by the relevance of his points to my own interest both in **innovation** and in understanding how collaboration technologies and social media can **support project management**. Following an outline based on Stein's paper, I discuss some of these comparisons below.

1. Software development is not the same as product development.

Stein distinguishes between *software development* and *product development*.

In product development, Stein says, innovation is frequently desired and rewarded. Risks are involved, but tolerated. A software development project, even when run using an agile approach in place of a more traditional "waterfall" model, is more concerned with managing and reducing risk than with creating and inventing new requirements and features.

According to Stein, when agile techniques are employed in a software development project and a series of "timeboxed" short-duration development periods are followed in order to maximize releases and user feedback, research and experimentation are minimized.

Stein does say that innovation in the process of *managing* software development using agile techniques may actually be promoted, even if this does not carry over to the delivered software itself. Agile effectively reduces risk by focusing on only the most high priority requirements in

short term time periods and on maximum responsiveness to frequent client review.

While there is no doubt in my mind that collaboration tools and social media would be useful in a product development effort that spans many individuals and locations, its applicability to a tightly controlled agile software development project, especially one involving a small self contained team, may be questionable.

The tools I'm referring to here include enterprise-secure blogs and wikis, collaboration tools with workflow and group collaboration features, group chat and messaging, document sharing, "people pages" and search features that simplify tracking down and locating expertise, and bookmark and link sharing tools.

Use of such tools in an agile software development environment may prove to be a distraction if they are used randomly or not in a fashion consciously directed towards the software under development. They might also detract from the one-on-one and face to face meetings favored by many agile practitioners. For example, given the controlled access an agile framework provides for user review of incremental releases, adding more opportunities for the client to "lean over the shoulders" of the developers might actually prove distracting.

On the other hand, use of such tools among the developers themselves might actually strengthen the development process by focusing more efficient attention on the actual development process; this might reinforce what Stein says about agile's support for innovation in the management process itself.

Use of collaboration and social media tools in support of product development, where risk is tolerated and creativity and innovation are promoted, might be more appropriate, given the ability such tools have for improving collaboration and for promoting the fast and efficient exchange of ideas among a potentially larger group of people.

Of course, not all product development efforts involve innovation and creativity. Some product development efforts are "me too" exercises designed to fill out a catalog or product line for standard or commodity type products. Even there, however, when product features are not innovative, the process by which the product is developed (and manufactured, distributed, and supported), can be the focus of collaborative efforts aimed at creatively reducing or controlling costs as a key competitive weapon. Social media and networking tools can support such collaboration.

2. Agile practices emphasize risk reduction, not innovation.

Stein lists the following four elements of agile software development and emphasizes how they reduce risk:

1. Timeboxed iteration (encourages task completion in a defined time period)
2. Prioritization of requirements (ensures that high valued features are addressed first)
3. Concurrent engineering
(when possible, tasks are performed in parallel not one after the other)
4. Timely, invested customer/end user feedback (helps ensure that the software does what the client needs it to do)

Stein takes each of these agile characteristics and shows how they actually work against innovation. He also makes the point that controlling variability in a process is something that is very important in a manufacturing type of situation. Since innovation increases variability and uncertainty, removing the need (or opportunity) for innovation improves process repeatability and control.

While I agree with what Stein says with respect to software development, it is also useful to point out that collaboration technologies and social media are relevant both to process oriented as well as innovation oriented operations. Collaboration, which I define as “people working together to achieve a common goal,” occurs in a wide range of operations, not just in white-collar, knowledge intensive, creative, or artistic endeavors such as software development.

Collaboration is performed by assembly line workers, call center representatives, and repetitive document processing and coding operations. Even though the fundamental activities of these operations may be highly repetitive, the social relationships and communications surrounding how they are performed are an important mechanism for training, dissemination of changes, solicitation of feedback, and performance evaluation.

In other words, even though there is an apparent lack of opportunity for innovation in the performance of basic “manufacturing” or process-type operations, there is still a workgroup need to share information, especially if there is constant turnover in a workforce. Collaboration tools and social media, I would argue, can be just as important in these “repetitive process” situations as in more “knowledge based” activities that require and reward creation and innovation.

3. Innovation can occur within the software development process itself.

Stein makes the point, “Regardless of what is being built, the process itself can be incrementally improved each iteration.” In other words, even though the software being developed may not turn out to be innovative, the developers may have freedom in the context of the agile process to make continuous changes to the way the development process itself is managed.

This parallels what was said in the previous section. Even though the underlying product (software) or repetitive process (e.g., call center operations) may itself “resist” innovation due to various controls, there might still be opportunities to adopt and use collaboration tools and social media in support of the management process itself.

How realistic this is would be at least partly dependent on the scale of the project or operation. If agile software development is being employed in a small team setting consisting of people all located within the same area, there may be no need or opportunity to adopt additional tools to support improved collaboration.

If the team is large or is distributed across multiple locations, departments, and functions, collaboration tools and social media might make good sense to improve communication and collaboration effectiveness.

4. Employ concurrent engineering to reduce development times.

Stein suggests that doing work in parallel — for example, simultaneously developing both infrastructure and tools as well the product itself — can save time. That’s hard to argue with, as long as the cost of managing dependencies across concurrent tasks can be efficiently managed.

The benefit of tasks performed in parallel is not unique to agile project management. It’s not unusual to optimize a traditional “waterfall” type project schedule by implementing parallel tracks where possible. In both cases, using both management attention as well as collaboration tools to keep teams synchronized instead of waiting till the end of a time period to check performance seems to make sense. Modern tools such as instant messaging, web based conferencing, and even mobile tools that track geographic location of smartphone users can contribute to continuous cross-team communication and collaboration.

5. Separate innovation-driven from production activities.

This is another example where agile and traditional waterfall project management techniques are similar. Anyone who has managed a large or complex project will understand this need to adapt management techniques. Some high-risk tasks may require creativity and innovation. Other low risk tasks in the same project may require mind-numbing repetition. The people, processes, and systems associated with these tasks may need to be managed differently.

Because variations exist in the types of controls appropriate for different types of tasks (e.g., concepts such as “milestone” and “deliverable” may differ) I would suggest that, at minimum, the project’s communication infrastructure should be unified and transparent. People should have one place they can go to to view, discuss, and report on their own set of tasks, they should be able to see what others are doing, and they should be able to put their own work into an overall context that allows them to see how what they do impacts “the big picture.”

The value of this type of transparent communication infrastructure is one of the reasons I have been researching the adoption and use of tools such as [blogs as project management tools](#). Such tools can supplement specialized project management tools and techniques by making it easier to publishing information centrally and by making it easier to engage in conversations and discussions that are visible to all. (An added bonus is that collaboration tools and social media can reduce dependency on email and meetings.)

6. Don’t re-invent the wheel.

Stein emphasizes the need to make every feature of software justify its existence; don’t waste time developing features that people don’t need, he says.

This argues, he says, for close collaboration with all important stakeholders, not just those who represent the hands-on users of the application being developed.

In my opinion, this is one of the great potential applications for social media and collaboration tools in a project management environment: expanding the sources of feedback in a controlled environment. While there is no substitute to sitting down with a “real user” and walking

through how an application is used, the ability to open up and engage in discussions with a wide ranging groups of stakeholders seems an ideal way to make sure that only needed features get developed.

7. Optimize the length of the release cycle.

In agile project management the length of the release cycle is critical. Make it too short and not enough useful functionality will be provided to users for the all important feedback. Make it too long and a tendency for “feature-itis” might start to creep in.

Of key importance, according to Stein, is establishing the release cycle so that it is synchronized with the team’s creative processes, not just with the needs of the overall production process. Another consideration is the need to synchronize performance across groups.

What the project team must decide is how long it will take, given project resources, to create a release that provides a measurable increase in functionality. This functionality should be significant enough from the user’s perspective to constitute a meaningful difference from the prior release.

While there there may be no simple formula for calculating what constitutes an optimal release cycle length, Stein does suggest that defining cycle length is the product of considering many factors. If as suggested in the previous section using collaboration tools and social media is one way to increase involvement of more stakeholders in the project, perhaps we might also be justified in asking whether to consider modifications in the definition of a “release” that takes into account the continuing popularity of “beta ” releases being so widely — and permanently — available on the web. In other words, why wait until the end of a release cycle to bring in the users?

Why not make continued release an ongoing process where user participants can dip in and provide feedback at any time without waiting till a formal release version has been decreed?

OK, I’m half kidding. I understand how critical timeboxing, release cycle, and user feedback are in the agile project management world. Purposely throwing away the concept of successive releases by making ongoing — and potentially minimally differentiated — versions available in an ongoing basis sets this agile formalism on its ear. But it would be one response to the realities of how public the software development lifecycle has become in our “web 2.0” world. It might also be a way to accommodate a project that has established the need for innovation and its resulting difficulty of defining, at least early on, how long the release cycle should be.

8. Define project off-ramps.

Stein distinguishes between an agile project’s normal deliverables and assessment checkpoints where a decision is made whether or not to continue with the project. He points out the need to communicate to staff and management the difference between the two.

Such go/no-go decisions are not unusual in traditional waterfall type projects, especially in projects where a long or complex series of steps must be accomplished to reach a desirable end point. An example of such a project would be consolidation of two large databases where there

are many intermediate steps that must be accomplished in order to convert data from a source to a target format. If part of the conversion process fails the entire project may be stopped since there is no sense going forward with faulty data.

The comparable situation in an agile project would be where a decision to initiate the project was made even though uncertainty existed about project success without development of significant creative or innovative solutions. Relying on the ability of the project team to create such innovative approaches could be thought of as a high-risk situation that required periodic checkpoints to formally decide whether or not proceed.

Regarding how to apply collaborative tools or social media in such a situation: shutting down a project can be a traumatic situation for all involved. Making the go/no-go decision criteria known to the project team is a communication objective that can be accomplished a variety of ways. Collaborative tools could also be used to allow project staff to participate at some level in the decision of whether or not to proceed.

Observations

I have some personal experience with collaboration systems, social media, and — as a self-taught project manager — with both “traditional” and “agile” project management approaches. I’m a firm believer that project managers need to select techniques that fit the needs of the client and the project. This includes mixing and matching agile and waterfall techniques within the same project, if that makes sense.

I also know it’s easy to get caught up in the almost religious arguments that surround project management methodologies. Let’s just say, therefore, that I’m a pragmatist when it comes to managing projects. That said, here are some observations:

1. Projects that are “pure software development” are rare.
2. Project management skills should not be limited to the IT department.
3. Collaboration systems and social media should be standardized and part of the infrastructure.

1. Projects that are “pure software development” are rare.

In his article Stein distinguishes between “product development” and “software development” projects. While I agree they are different, in my experience in project management and consulting I have found that “software development projects” usually, if not always, consist of more than just “software development.”

Especially when it comes to implementing a new application within an organization, overall success (and project completion) are usually also dependent on staff hours devoted to non-development activities such as communications, management oversight, training, quality control, and other business and management related activities.

This is especially the case in “web 2.0” situations where a remotely hosted application is not developed in-house but is procured as a service. In such situations the cost of the software may be insignificant compared with the other projects. While software-as-a-service evangelists may wish to soft-peddle this reality in order to minimize the true costs of implementing an

application and its associated business process changes, management still needs to take into account the practical realities and costs of reaching a point at which the new system is helping to generate real benefits.

2. Project management skills should not be limited to the IT department.

In my experience the project management skills of the IT staff usually outweigh the project management skills of non-IT staff. This is true even in situations where a new software project involves significant business process changes within the client business unit.

Recently I've noticed availability of an increasing number of software tools, many remotely hosted, that provide certain project and task management capabilities that were previously only associated with "formal" project management software applications such as Microsoft Project. While some of these tools might be considered "lightweight" from the vantage point of a grizzled IT project management veteran [you know who you are], I see great benefits to extending access to project management tools and concepts beyond the IT staff into business areas .

Given the IT department's expertise in project management, I would suggest that one possible role for IT should be to assist business units in learning and using project management tools and techniques. That should include not only training users on formal tools such as MS Project as well as more user-friendly tools such as the "lightweight" tools I referred to earlier, but also when to seek out "high ceremony" versus "low ceremony" project management approaches such as traditional waterfall or agile techniques.

3. Collaboration systems and social media should be standardized and part of the infrastructure.

The last thing a fast moving project team should have to do is spend time learning a new or unique communication tool just to support a single project. This is one of the reasons I believe that collaboration systems and social media tools should be standardized and generally available to any group within the organization that needs collaboration support.

Such tools should also integrate well with the specialized approaches and applications used within the project in question. The ability to make such an assessment calls, I believe, for direct involvement of the IT department, even if the tools are being remotely hosted.

- *Copyright (c) 2008 by Dennis D. McDonald.*
- *To comment on or discuss this article email the author at ddmcd@yahoo.com or use the form below.*

Article originally appeared on Dennis McDonald's Blog (<http://www.ddmcd.com/>).

See website for complete article licensing information.