Are Management Basics Affected When Using Agile Methods?

Paul E. McMahon
PEM Systems

Just how different is project management when using agile methods? The purpose of this article is to help readers understand the similarities and differences of traditional and agile project management approaches, as well as provide information that can help them decide if an agile – or a hybrid agile – approach might be beneficial. Related factors to consider when making decisions about using Agile, hybrid-agile, or a traditional approach, along with real project case studies, are provided.

Let us start with the basics. First, fundamental to project management is planning, monitoring, and controlling. Monitoring and controlling are achieved by executing a plan and taking appropriate action when a project deviates from that plan. This article focuses on the planning activity. I like to simplify planning for new managers by breaking it down into five easy steps: *What? Who? When? How?* and *How Much?*

Traditionally, a project management plan is developed at the start of a project to capture the answers to the first four questions. The plan is then used in the *How Much?* category to develop the cost, monitor and control the project, and communicate to the stakeholders what we are doing.

At the fundamental level, planning includes understanding *what* must be done (scope of effort), *who* needs to do it (staffing and skills), *when* it needs to be done (life cycle and schedule), *how* it is to be done (reviews, methodology, tools, meetings etc.), and *how much* it will cost (budget). These same five steps occur on both traditional and agile projects.

The *What*: Scope of Effort

Traditional Approach

Project planning includes scoping the work. Traditionally, this has been accomplished by first partitioning the effort through a work breakdown structure [1]. The intent is to break down the work into manageable chunks that can be monitored and controlled.

Figure 1: *Traditional Functional Approach*

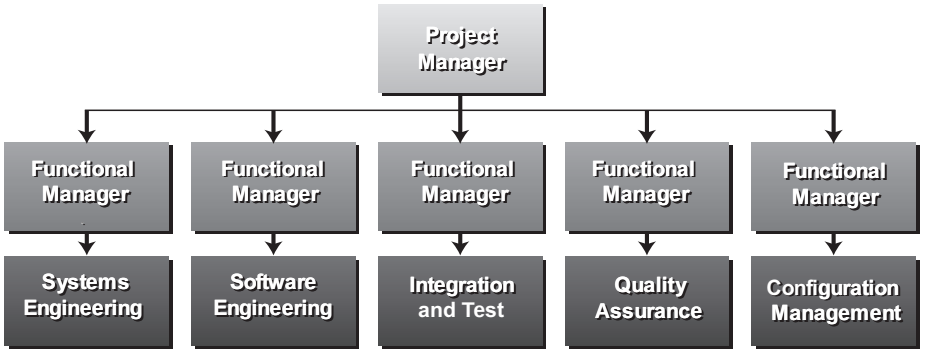
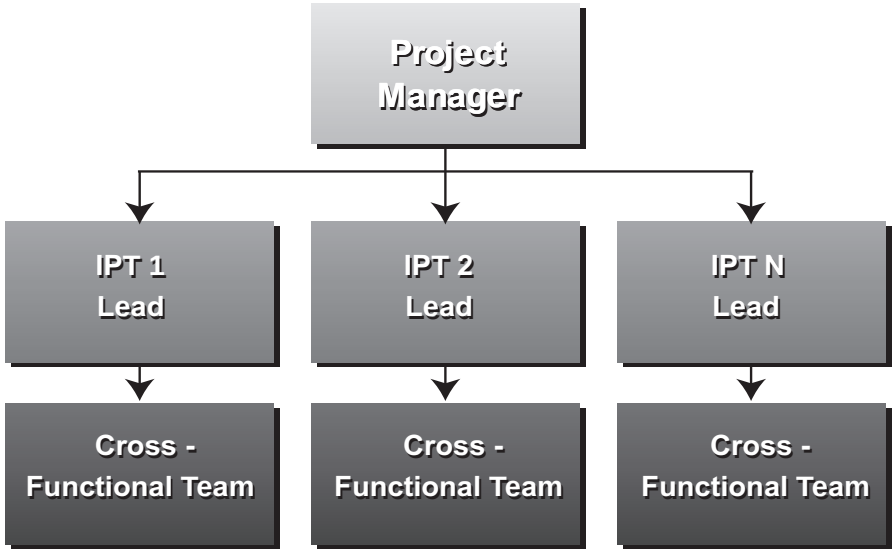


Figure 2: *Traditional Integrated Product Team Approach*



Agile Approach

The basic concept of breaking work down does not change with agile methods, but there is less detail provided early in the project and care must be taken in how work is structured so that it is fully scoped while potential solutions are not overly constrained. Some agile projects do not fully scope all the work up front. Scoping the work is discussed further in the section on *The When*.

The *Who*: Staffing, Skills, and Organization

Traditional Functional Approach

Traditional functional engineering organizations include systems engineering, software engineering, integration and test, configuration management, and quality assurance (see Figure 1). In the traditional, functional organization, tasking is through the functional manager, and the functional manager receives periodic status directly from assigned personnel.

Traditional Integrated Product Team Approach

Integrated product teams (IPTs) are cross-functional teams used in many organizations to achieve increased stakeholder collaboration and teamwork. Each IPT includes representation from all functional disciplines. Historically on large projects, IPTs have often been large teams (e.g. could have between 30 and 50 people on each IPT); therefore sub-IPTs may be formed for specific tasks. IPT tasking is usually through both the functional manager and the IPT, with the functional manager providing a higher level task definition, and the IPT providing project specific tasking (see Figure 2).

Agile Approach

The traditional functions are all still required when using agile methods, but the organization of the personnel, their interactions, and task reception may be different. The degree of difference depends partially on how your organization currently functions. Agile teams are small (usually no larger than 10 people each), and they are cross-functional like the traditional IPT. Two views of an agile organization are provided in Figures 3 and 4.

Figure 3 provides Agile Organization View 1, which has similarities to the traditional IPT structure. With this structure, agile teams operate as an extension of the traditional IPT similar to sub-IPTs in traditional organizations. Note the following caution associated with this view.

Not all organizations today implement true IPTs in the sense of *true* cross-functional and product focused teams. Many organizations have struggled with implementing IPTs because of the difficult culture-shift required from a functional perspective to an integrated product perspective. Similar issues are faced when implementing agile teams which are also cross-functional integrated product teams. Typical differences with agile teams from IPTs are their size (smaller), incremental approach to work, and visibility of tasks and task progress.

While Figure 4 appears significantly different from Figure 3, the real difference within a given organization may be more pictorial than real. Some have argued against representing agile teams as sub-teams of IPTs because it may give the impression they operate in a traditional, hierarchical, and functional fashion where tasking only flows down from management.

The *Hub* organizational structure [2, 3] depicted in Figure 4 is intended to represent the fact that the team is actively involved in its own task definition and estimation process, and proactively communicates with other teams directly when necessary. This is in contrast to the traditional, hierarchical organization where the focus of tasking and communication is through the chain-of-command. It is worth noting that this perception of how communication happens in traditional, hierarchical organizations is not always true.

This leads to the question: How different is the hub organization from the traditional organization? The answer to this question depends largely on the culture within your organization today. While the traditional organization structure represented in Figures 1 and 2 may seem familiar to many readers, implementations within specific organizations can be significantly different [4].

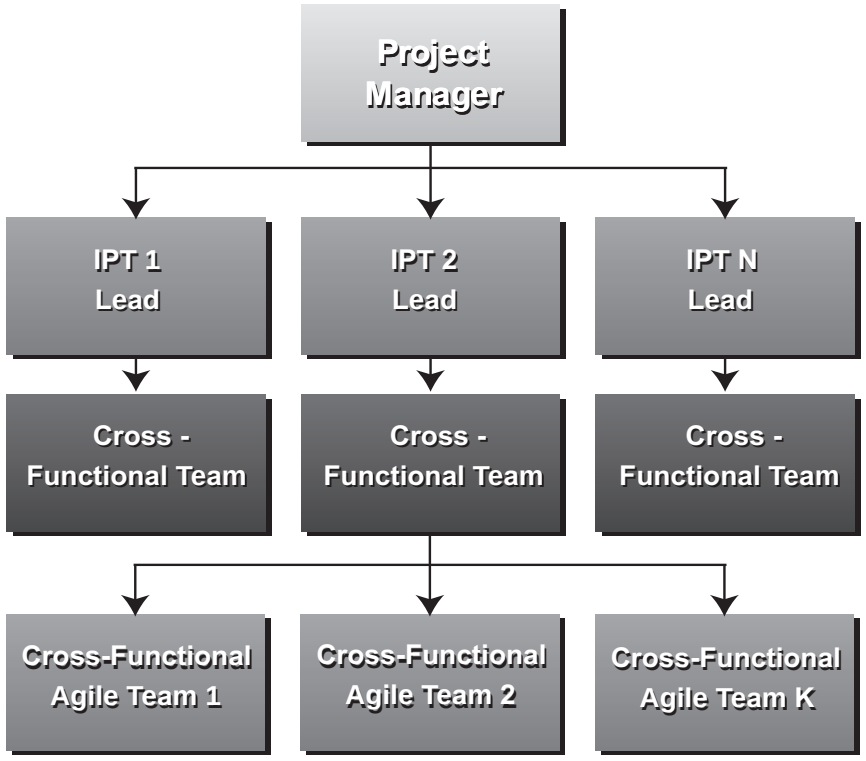


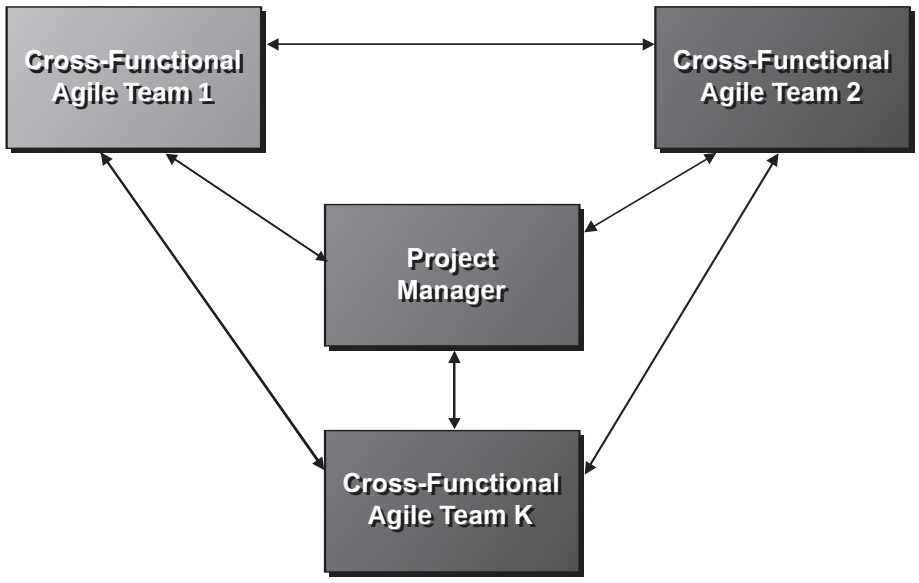
Figure 3: *Agile Organization View 1*

If an organization has effectively implemented *true* IPTs, and functional managers already have shifted their tasking to a higher level in support of project specific tasking by the IPT, then the shift to Agile may not be traumatic from a management perspective. This is because some of the hardest changes with Agile are cultural. If the culture is already *agile*, this makes the transition easier. In many organizations this operating model has existed for many years informally [4, 5].

Case Study 1

I have observed this condition in one of my client organizations that does not even refer to itself as being agile, yet they have exhibited agile-like behavior for many years. I have referred to this in previous publications as an unspoken adaptive (agile) subculture [5]. In this organization, small informal teams operate below-the-radar of the formal organization with a do whatever it takes team attitude to get the job done. Tasking from the functional managers in this organization is at a high enough level so that it is not an issue for a software engineer to help a systems engi-

Figure 4: *Agile Organization View 2*



neer and vice versa. It is expected as part of the job. The culture of what it means to do software engineering includes collaboration with systems engineering. I have referred to this in previous publications as the integrated tasking model [4].

Case Study 2

On the other hand, I have another client who employs what I have referred to as a *strict hierarchical tasking model* [4]. In this organization, systems engineers are tasked strictly and exclusively by the systems engineering functional manager (see Figures). In this organization it is not uncommon for a systems engineer to write a specification in her private office and then effectively *throw it over the wall* to the software engineering group with little personal interaction throughout the process. When a software engineer finds a problem in a specification from system engineering in this organization, they are not allowed to change the specification and the culture does not encourage software engineers to go talk directly to a systems engineer with respect to a potential change. They have to send it back through the formal chain, and it takes forever to get changes approved.

Different Reasons and Challenges When Moving to Agile

The organizations for both case studies use diagrams that look similar to Figures 1 and 2 to represent how they operate today. Both are interested in moving their organizations toward more agile practices. But the reasons they want to be agile and the challenges each face in making this transition are very different.

Why Would an Organization Want To Be Agile?

The reason Case Study 1 wants to move to Agile is for better visibility of work status and increased predictability. While this organization already exhibits a small team collaborative attitude, decisions sometimes get made without considering all the consequences. As a result, negative side-effects are not uncommon, such as slipped schedules and build instability. They are constantly at risk of falling into chaos and often rely on late-night heroics by individuals. The reason Case Study 2 wants to become more agile is because changes take far too long to cycle through all the bureaucracy in the organization, and they know they are not being responsive to their customer's needs.

Management Challenges Faced

In Case Study 1, a key management challenge is to get the small teams to self-manage their work more effectively. In Case Study 2, a key management challenge is to move decision making down in the organization and increase collaboration at the lower levels.

The *When*: Life Cycle and Schedule

Traditional Approach: Work Partitioning

Traditionally, work is partitioned into major functions, and those functions are further partitioned into tasks associated with requirements, design, code, test, and integration. Detailed schedules are developed up front in the project and task assignments are given to each developer.

Agile Approach: Work Partitioning

While the traditional functions are all still required, when they are done may be different. Up front, coarse grain planning is done [6]. This is high level planning for the long term of the project. This includes the high level requirements which are then allocated to increments. This is similar to traditional incremental planning, but with Agile, more details may be deferred which implies more planned collaboration with the customer later in the project.

As an example, one of my clients is modernizing a legacy system. We defined all the legacy system functional requirements up front, but we deferred details of the user interface. The most important thing to the customer was to move the functionality to the new system and shut down the legacy system as soon as possible. We did identify high level user interface requirements early as part of our coarse grain planning, but we deferred details because it was not high priority to the customer.

The customer collaboration on the user interface details needed to be planned. This collaboration was so important that we added it to the project schedule. This last step was also important to aid accurate progress reporting. Planned and scheduled collaboration throughout the project was a key difference with Agile methods.

Why Defer Details? Potential Advantage

The rationale for deferring details is based on the belief that by waiting to make decisions just in time, we have the best information possible and therefore reduce the chance of rework. Reducing rework means cost savings. Also, as we saw in the example of the legacy system, we may defer work based on value to the customer.

Why Defer Details? Potential Disadvantage

While on the surface this may seem logical, deferring details has also been known to lead to inaccurate progress reporting and scope creep late in projects. This is why it is important to place deferred work on the schedule or team task list where it is kept visible.

Recommendation to Help Minimize Scope Creep on Agile Projects

One recommendation that I have previously made [7] to help control requirements on an agile project is to fully scope the requirements up front at a high level (as we saw with the example of the legacy system) and then plan and schedule the time to work out the details at the start of each increment. If you have a good collaborative relationship with your customer this extra level of requirements control may not be necessary. But my experience on United States Department of Defense contracts has found more of an *adversarial customer-contractor relationship* than a collaborative one and therefore this extension to Agile for requirements control seems practical.

Agile Approach: Tasking and Scheduling Responsibility

While project scheduling and personnel tasking are still required with Agile, where responsibility lies and when a task is done may be different. Before making any scheduling and tasking responsibility changes, look closely at what your organization is doing today and analyze the effectiveness of the current process.

In some organizations, the traditional approach to scheduling is to build a large detailed schedule early in the project. The problem with this is that it can become difficult to maintain when changes on the project happen quickly. When this happens, senior management may no longer have an accurate picture of where the project truly is from a schedule perspective. If your schedules are accurately reflecting your project work, and you are able to keep them current, then it may not make sense to consider changing what you do. But if they tend to be difficult to maintain and often out of date then this may be a good area where some agility can help.

How Agile Can Help With Schedule and Task Status Accuracy

By keeping the project schedule at a higher level it becomes easier to maintain. This is not to say the details are not important. But by placing the details down inside each agile team and giving the agile teams the responsibility for keeping their status visible and up to date, one may be able to

increase their chances of accurate status reporting on projects.

A question often asked by managers considering moving toward Agile is *how difficult is this change going to be for my organization?*

The answer to this question usually depends on where your organization is today. In Case Study 1, the challenge faced was to get the team leaders trained in how to manage small teams more effectively and letting the team members know they are each responsible for maintaining and reporting their own task status back to the team. In this organization, there already existed some very successful small team leaders. The challenge was to mentor others in what the successful leaders in the organization were already doing so more of the organization could benefit.

Example of Management Change When Transitioning to Agile

In Case Study 1, the organization is gradually learning they need less detail in the high-level schedule as the small teams provide increased visibility into their task status. As more small teams in the organization meet their commitments, senior management's confidence grows, and they start to ask for less detail at the top. The change is not yet complete, and it is not happening over night.

Traditional Approach

Traditionally, when setting up work packages and planning the work to do, it makes good sense to use what has been referred to as a *rolling wave* approach. This means only plan work in detail for a short duration. When this short duration period gets close to the end, then plan the next wave in detail.

We used to do this 30 years ago when I was a programmer and a manager working for government programs building aircraft simulation systems. The *rolling wave* approach to planning is consistent with agile thinking today – plan the details incrementally and *just in time*. But over the years in some organizations, culture and control-oriented managers have pressured engineering organizations to plan excessive detail early.

Agile Approach

Agile approaches are driving us back to execute the rolling wave concept as it was always intended. But doing this the right way is not always easy, which is part of the reason why managers seeking to control their projects pushed for more detail early. When we do not plan the detail early, it becomes easy to abuse the process by pushing out real work that should be going on now. This potential downside of Agile needs to be taken into account when choosing whether to go with an Agile or a traditional approach.

The *How*: Tools and Motivating Teamwork

Traditional Approach

Traditionally, when establishing a plan up front, reviews to be conducted with the customer and internal reviews to the organization are identified. The methodology to be used is also established, along with planned tools.

Agile Approach

With Agile, reviews, methodology and tools must all still be planned, but the focus of the team shifts from the tools and methodology to personnel interactions concerning real project status. Tools can also affect the accuracy of status reporting as seen in the following case studies.

Case Study 3

At one of my client locations, Scrum [8] (a popular Agile method) was being used on a number of projects. The project leads and developers were reporting positive results with improved status reporting. The Sprint Backlog (Scrum term for team task list) was kept informally as *sticky* notes on the walls in conference rooms. Thinking it would improve the process, a commercial task management tool was introduced in the organization. Soon thereafter, team enthusiasm for the process and the disciplined and accurate status reporting fell off. After doing a little digging, I discovered that the commercial tool that had been introduced as a *process improvement* was not easy to use. The developers viewed the tool as a burden, which led them to stop updating their task lists and related progress in a disciplined way.

It is easy to look at this case study and say *tools can be fixed*, but too often they are not, which can result in inaccurate status reporting. But beware – there is another side to this story.

Case Study 4

A common practice on agile teams is to have team members sign up for work, rather than being directed to work on a task. The rationale for this practice is the belief that it promotes personal commitment to completing the task more effectively and on schedule.

My client, who does not refer to itself as agile (Case Study 1), uses a tool for task management that many in the organization do not like. All the developers in the organization are required to log into the tool every day to get their current assignments and report their status. People continually complain about the tool because of the time it takes to use it.

I would not say this organization applies self-directed team practices as described in many agile books, but they may still achieve a good part of the intent.

As an example, individuals are given tasks through the tasking tool by functional managers. They do not *sign up* for each task, but individual task performers do have an opportunity and are encouraged to communicate back with their manager after they have analyzed their task. If they do not feel they can meet the task due date, or they feel the allocated hours are insufficient, or if the task description is not clear, they can send the task back to the manager.

This communication back and forth usually creates healthy task discussions, driving an understanding of the real work being faced. As a result, increased visibility of where projects truly are is observed and senior management becomes aware earlier when projects are getting into trouble. There is also an improved customer confidence that has been observed as well by many throughout the organization.

It is worth noting that this organization-wide tool makes it easier for project team members, who are not collocated, to get their tasking and report progress, as well as participate on projects as team members from remote locations. It is also worth noting that sticky notes on conference room walls do not scale well especially on large distributed projects.

Motivating Teamwork

Because people sign up for tasks, and are asked to help others, an argument against self-directed teams has been, *why should I do my job and yours too?*

Jeff Sutherland, co-founder of Scrum, provided one idea how to do this through a performance review process he applies within his own company. It is a *weighted average individual performance review process* where the rating components include inputs from the customer, the team, and the company perspective. With this approach, employees can no longer get good reviews based only on the perceptions of their functional managers. Their review now depends on what their customer and teammates think, as well as their overall contribution to the organization. This technique can be used to help teams collaborate more effectively in both agile and traditional environments and it could help whether tasks are

assigned or *signed up for*.

The *How Much*: Cost and Metrics

Agile Cost

We do not yet fully know how cost is affected by employing agile methods. There is not yet enough real project data to draw conclusions. But with Agile it is a mistake to think it will cost less because we do less engineering. Rather, we partition the engineering work and do it at the best time, which should reduce rework cost. This can be done with hybrid agile methods and traditional methods as well.

Agile and Traditional Use of Metrics

Some organizations (both agile and traditional) use burn down (or burn up) charts [9], to indicate schedule progress. In organizations that follow Agile strictly, burn down charts are owned by the team, not functional managers. It is not a separate manager's view. But there is another side to burn down charts. Sometimes an objective perspective from outside the team can help, especially when team members lack progress estimation experience.

Ask yourself, who owns the burn down charts in your organization? Is it the team's perspective, or a separate manager's perspective? Is someone filtering the team's view? This is not necessarily bad. It might actually help convey the real progress more accurately, but it might not. It depends on your company's culture and project-specific conditions. But one indicator of whether it is working well for you is the accuracy of the reporting up the chain. Are you hitting your schedules? Do you have satisfied customers? Ultimately, these are the questions that you should ask to determine if your metrics and your status reporting system are working for you.

With Agile, the team members report their progress on the tasks they sign up to do. In *hybrid-agile* organizations such as Case Study 4, if they are given the task, they are also given an opportunity to discuss it with their manager and refine it, if necessary. When this process is working right, each day you can see the work to do – or team task lists – being updated, and work accomplished being checked off.

Agile Tailoring for U.S. Government Projects

Those who follow Agile strictly report progress on their task lists when code has been tested and works. In my recommendations to contractors on government projects, I tailor this strict software reporting focus to add all real tasks including documentation and preparation for customer reviews. My rationale is fueled by a desire for the burn down chart to represent *all* real work. When the burn down chart hits zero, I want to know that we are *really done*.

Agile and Traditional Use of Lines of Code Metric

Agile does not use lines of code as a measure of productivity, nor do many organizations using traditional methods. The problems with using lines of code as a progress (or productivity) indicator have been well documented, and with Agile this does not change.

First, this measure tells us nothing about value to meeting the customer's needs. It just tells us we have generated code. With Agile, the focus is on doing the simplest thing to achieve customer value, and it is viewed as more valuable to achieve it with less.

On the other hand, I have seen lines of code metrics used with traditional approaches as an effective trend indicator. For example, tracking the number of lines of code changed or added from one build to the next is a useful trend indicator, especially as a team nears a delivery milestone. It gives us a view of potential build stability and another perspective on how close we may really be to completion. The use of lines of code metric in this way is no different for an agile or traditional project.

Conclusion

The five basic steps of planning and controlling a project remain, but when employing Agile methods, these steps may be carried out with some key differences. Agile methods provide the opportunity for more accurate project status through self managed teams, and they also provide the opportunity for more rapid change processing. Case studies indicate that hybrid Agile-traditional approaches are often appropriate and can be particularly effective when based on the culture of a given organization.

Just how different project management is when using agile methods depends on the organization. It is not a matter of being agile or not being agile. There are many degrees of agility, and one can anticipate many decisions to be made along the way.◆

References

1. Project Management Institute (PMI). A Guide to the Project Management Body of Knowledge (PMBOK Guide). 3rd ed. Newtown Square, PA: PMI, 2000.
2. Highsmith, Jim. Agile Project Management. Addison-Wesley, 2004.
3. McMahon, Paul. "Lessons Learned Using Agile Methods On Large Defense Contracts." CROSSTALK May 2006 <www.stsc.hill.af.mil/crosstalk/2006/05/index.html>.
4. McMahon, Paul. Virtual Project Management: Software Solutions For Today and the Future. CRC Press, LLC, 2001.
5. McMahon, Paul. "Integrating Systems and Software Engineering: What Can Large Organizations Learn From Small Start-Ups?" CROSSTALK Oct. 2002 <www.stsc.hill.af.mil/crosstalk/2002/10/index.html>.
6. Cockburn, Alistair. Crystal Clear: A Human-Powered Methodology for Small Teams. Addison-Wesley, 2005.
7. McMahon, Paul. "Extending Agile Methods: A Distributed Project and Organizational Improvement Perspective." CROSSTALK May 2005 <www.stsc.hill.af.mil/crosstalk/2005/05/index.html>.
8. Schwaber, Ken. Agile Project Management With Scrum. Microsoft Press, 2004.
9. Cohn, Mike. User Stories Applied: For Agile Software Development. Addison-Wesley, 2004.

About the Author

Paul E. McMahon, principal of PEM Systems, helps large and small organizations as they move toward increased agility. He has taught software engineering, conducted workshops on engineering process and management, published articles on agile software development, and authored "Virtual Project Management: Software Solutions for Today and the Future." McMahon is a frequent speaker at industry conferences including the Systems and Software Technology Conference, and he is a certified ScrumMaster. He has more than 25 years of engineering and management experience working for companies, including Hughes and Lockheed Martin.

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