Rally Software Development Corporation

Whitepaper

A Project Manager's Survival Guide to Going Agile

Michele Sliger
Agile Coach, CSM, PMP
msliger@rallydev.com

030206

Abstract:

When software development project teams move to agile methodologies, they often leave project managers behind. Traditionally trained project managers are confused as to what their new roles and responsibilities should be in an environment that no longer needs them to make stand-alone decisions. This paper focuses on re-defining the job of project manager to better fit the self-managed team environment, one of the core agile principles. Special emphasis is placed on the shift to servant leadership, with its focus on facilitation and collaboration. Mapping of PMBOK knowledge areas to agile practices is discussed at length. After reading this paper, project managers should have a better understanding of what changes they need to make professionally, and how to make these changes in order to survive the transition to an agile software development approach.

Table of Contents

The Movement to Agile Methodologies

Agile software development methodologies such as Extreme Programming (XP), Scrum, Crystal, Lean, etc., are becoming more prevalent in the industry. What is it about these approaches that have companies all over the world casting out their traditional plan-driven methods in favor of an agile process? And how do project managers fit into the new archetype, which favors self-managed teams?

Companies are moving to agile processes because the technology marketplace demands its suppliers be highly responsive to change. In order to compete in the global economy, companies must move quickly to provide solutions to a client base that has more and more choices available to them. Agile approaches promise faster delivery of working code, higher quality, and an engaged development team that can deliver on its commitments. Traditional waterfall, with its long phases and heavy investment in "big up-front design," lacks the flexibility to swiftly respond to the market.

It's also about the money. Agile processes can provide a larger return on investment by decreasing the investment in inventory, decreasing operating expenses, and increasing throughput (Anderson, 2004). In other words, by eliminating the time and money spent on designing an entire system – a design that may be outdated before it is implemented, a design that may have numerous pieces that are never even coded – and by eliminating the waste that accompanies heavy process, software development teams are able to provide rapid delivery of value to customers, resulting in greater profits.

Agile methodologies aren't going to be a fad. The transition to agile processes is a growing trend that will have lasting effects on the industry and the people involved. It's a different way to work, one that requires greater communication and cooperation from its participants and greater leadership from its managers. In fact, the job titles in use today – manager, director – are illustrative of the traditional command-and-control approach that are now giving way to the new guidance and mentoring method of working with teams. It will be a challenge for many of these managers to release control, as they begin to make the transition to being leaders.

Agile vs. Plan-driven Development

Both agile and plan-driven development recognize the triple constraint: cost, schedule, and scope. But whereas plan-driven development advocates locking down the requirements so that schedule and cost can be estimated, the agile approach says that scope is always changing and therefore schedule and cost should be fixed. This way projects don't become death marches, and the product is developed in a fashion where it's in a perpetual releasable state.

This basic shift in focus has a cascading effect on each subsequent practice in both approaches. Planning, executing, and controlling disciplines move from more directive, command-and-control tactics to facilitative, collaborative support. The idea being that if you give the team the tools they need, help them to understand the business problems they'll be solving, and give them the space and time to complete the job, that they can then be self-directed and self-organizing, and become a fully engaged and motivated team that produces high-quality products at a faster clip. Clearly, for many organizations, this change in tactics also leads to a shift in the culture and in the ways success is measured.

Project managers will find themselves learning how to guide their team in responding reliably to change instead of conforming to plan, and learning how to do this in a completely new environment where the team makes decisions instead of being told what to do. It means more individual responsibility for team members, and more facilitation skills required for the project manager. Most are uncomfortable with their new roles at first, and it is the project manager's responsibility to build team cohesion and foster good communication. Not everyone is willing to make this paradigm shift, project managers included. But for those who are willing, they'll find both the process and the new skills they've learned to be richly rewarding and well worth the effort involved.

Relating PMBOK Practices to Agile Practices

Ironically, for all the differences in Project Management Institute (PMI) and agile philosophies, many of the practices identified in the *Project Management Book of Knowledge* (PMBOK) are quite compatible with agile practices. Like the authors of the Capability Maturity Model (CMM), the PMI states that the PMBOK is a guideline of best practices, and organizations must use their own discretion as to how to implement them (PMI, 2000). In fact, agile methodologies, if followed with the discipline and rigor they require, are compliant with CMMI, just as traditional waterfall is compliant. What's different (beyond the basic command-and-control vs. self-managed teams) is in the when and how these practices are executed, and the new lexicon used by agile practitioners.

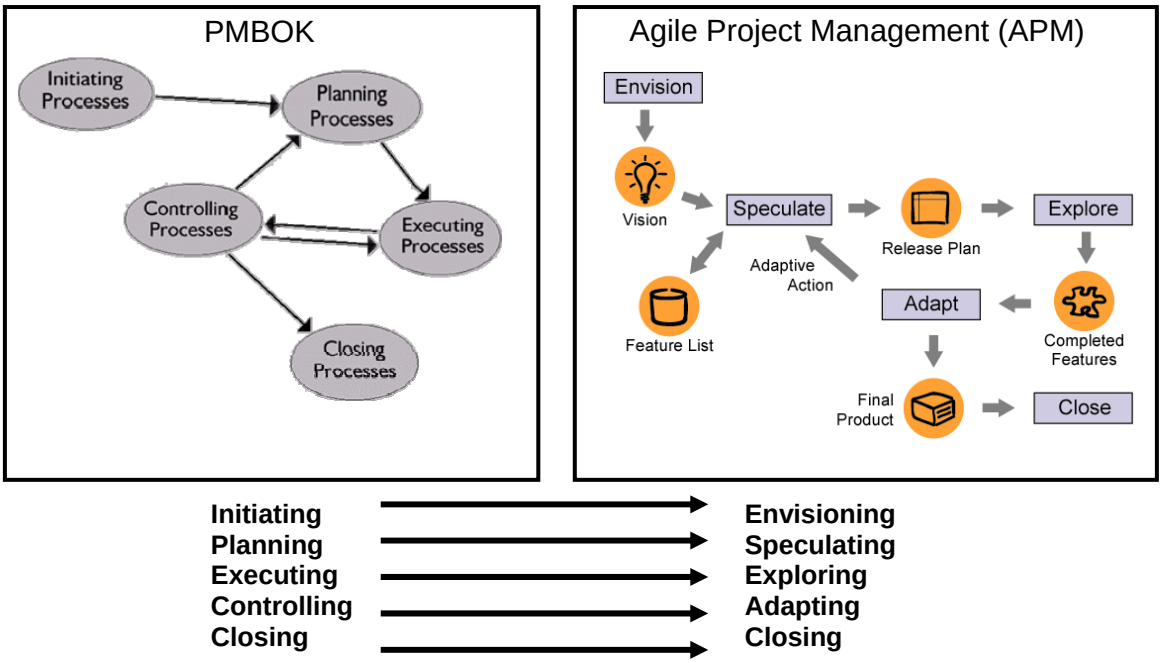

Figure 1:

PMBOK Project Management Process Groups mapped to Jim Highsmith's Agile Project Management framework.

The Project Management Process Groups

The first translation is from the PMBOK naming conventions for the project management process groups to Jim Highsmith's Agile Project Management framework (Figure 1). The PMBOK identifies Initiating, Planning, Executing, Controlling, and Closing as the process phases within project management. Highsmith reconfigured these phases to better reflect the reality of how software is truly developed, and defined them as Envision, Speculate, Explore, Adapt, and Close. The Envision phase is where the product vision is defined enough to provide a bordered sandbox within which to work. Then during Speculation, that vision is translated into a set of features and an expected timebox within which to deliver those features. The Explore phase is the iterative and incremental development of potentially shippable code, with planned stopping points along the way to inspect and adapt both the process and the product, as part of the Adapt phase. Finally, in the Close phase, teams have the opportunity to reflect on their achievements and make decisions based on what they've learned (Highsmith, 2004).

The project phases as described in the latest edition of the PMBOK also map nicely to an iterative development environment, scaling from short iterations up to longer-term releases (Figures 2 and 3).

A further translation of some of the actual activities performed in each phase is also warranted. Five key knowledge areas defined in the PMBOK deserving special attention are Integration Management, Scope Management, Project Time Management, Quality Management, and Human Resource Management. For each of these areas, the practices advocated by the PMBOK are significantly different in their implementation in agile software development.

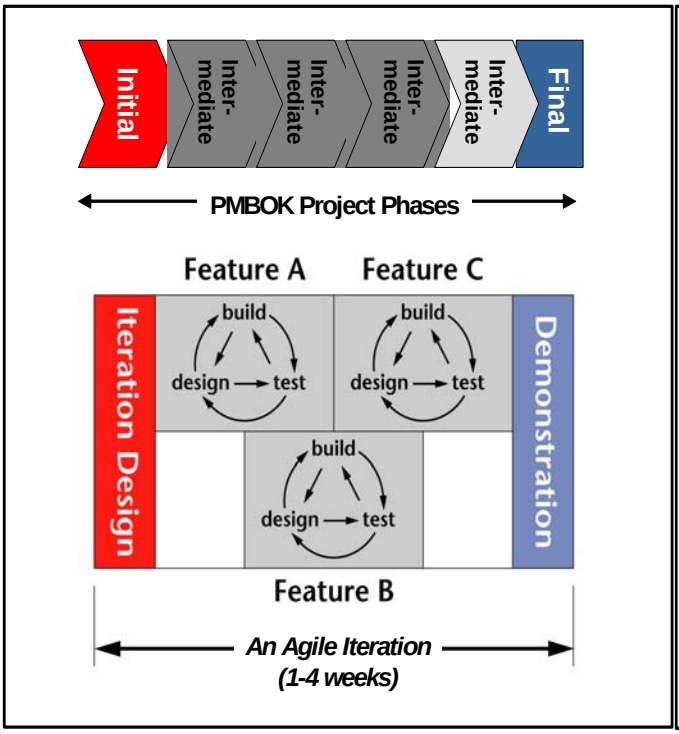

Figure 2.

PMBOK Project Phases mapped to an Agile Iteration

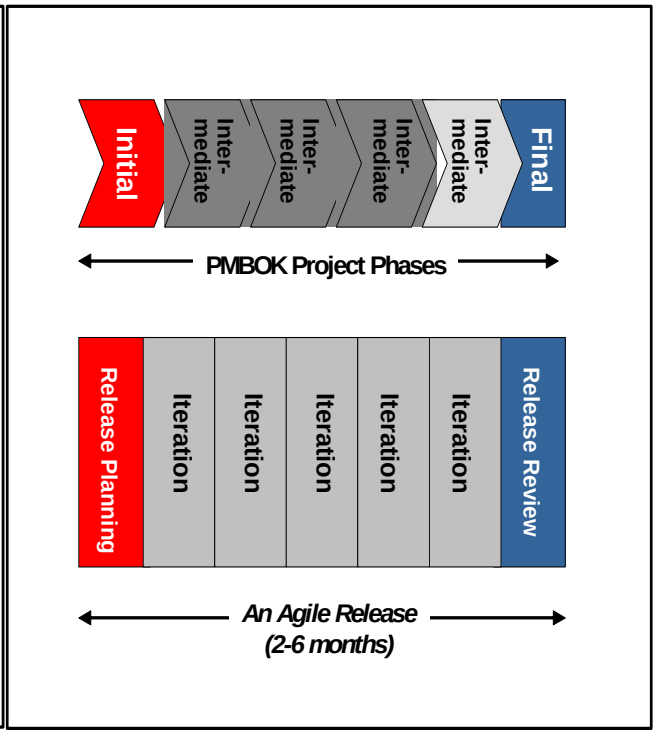

Figure 3.

PMBOK Project Phases mapped to an Agile Release

Integration Management

In Integration Management, a key deliverable is the project plan document prepared and owned by the project manager. In agile software development, with its emphasis on just-in-time design and participatory decision-making, this activity translates into several different envisioning and planning exercises done on an iterative basis. Rather than defining all of the elements of a project plan at the beginning of the project (scope, work breakdown structure, schedule, assumptions, controls) the agile project manager will instead focus on planning for the horizon, and low-ceremony documentation appropriate for the type and number of the communication nodes involved. Low-ceremony documentation of agile release and iteration plans can consist of several white boards in a war room with color-coded markings, or it can be flip-chart paper taped to the wall with brightly colored post-it notes on each.

(For help visualizing this, spend an hour watching Jim Highsmith's free webinar lecture on agile project management: http://www.rallydev.com/register_for_webinar_series.jsp#jimPresentation.)

Release plans indicating the expected release date and the features to be worked, and iteration plans indicating the level of effort in implementing a set of features within the defined timebox, are defined in planning meetings that involve the entire team. The team must create, own, and commit to the plan; they can not be ordered to perform to meet plan specifications written by an individual directing the project. Release plans, iteration plans, and other planning outputs from the Envision and Speculation phases are then shared with all stakeholders in the most highly visible fashion possible. For co-located teams, this may mean something as simple as posting the plan on the wall; for geographically dispersed teams, technical solutions

for communication are required, and SharePoint, wikis, or other third-party tools, including Rally, are well-equipped to provide this. Project managers go from writing a large detailed document defining the plan for the entire project, to facilitating the team in their ongoing iterative planning efforts and sharing that information in the most visible way possible.

Integrated change control also changes dramatically in agile methodologies. In keeping with the idea of minimum process to achieve maximum value, the change control process is streamlined and integrated into the daily routine of agile teams. Product change is managed via the ranked backlog of features. This product backlog is managed by the customer or customer proxy (product manager, subject matter expert, architect) who is responsible for maintaining the list of items to be worked, with the features that provide the most business value to the customer ranked highest. This backlog contains items beyond functionality requests: technical supporting work and defects are also placed in the backlog and ranked. During release and iteration planning, the highest ranked items move from the backlog into the iterations, to be coded, tested, and accepted by the customer. During the iteration, daily 15-minute stand-up meetings are held to apprise each other of status, a key communication element that keeps the team in synch and better able to tackle unforeseen issues. At the end of each iteration, the working code that was developed is demonstrated, and feedback is gathered from stakeholders that may affect future decisions about the items in the backlog and the ranking. Process changes are also made at the end of the iterations, allowing the team to make course corrections not only in the product, but also in the way they work. The team – customer, developer, tester, analyst, technical writer, project manager – becomes the equivalent of a change control board, streamlining the process so that decisions are made quickly, collaboratively, and with little to no ceremony.

Scope Management and Time Management

"Scope creep" has always been the bane of traditional project managers, as requirements continue to change in response to customer business needs, changes in the industry, changes in technology, and things that were learned during the development process. Scope planning, definition, verification, and control are all knowledge areas that are given great attention in the PMBOK. Agile methods believe these deserve great attention as well, but their philosophy on managing scope is completely different. Plan-driven approaches work hard to prevent changes in scope, whereas agile approaches expect and embrace scope change. Remember – the agile strategy is to fix cost and schedule, and then work to implement the highest value features as defined by the customer.

This is the area that tends to fluster project managers the most. The box that we use is still that of time; however instead of stuffing more and more feature "bricks" into a single flimsy box until the timebox explodes, we're now using multiple timeboxes made of steel, and we stop dropping in bricks when the box is full. We then close the box, padlock it for that iteration, and work the features through to acceptance. But because we're doing this box loading and padlocking one iteration at a time, it's difficult to understand how much work will be completed in a longer timeframe. In fact, the most common question asked by project managers, upon hearing of this new agile approach to planning and scope management, is "What am I supposed to tell my boss/customer when he asks when we're going to be finished?" Regardless of what approach you use, when asked to look that far into the future and provide answers, the role of project manager tends to slide into the arena of seers and fortune tellers. A project manager couldn't really answer the question definitively even using traditional project management approaches – it was simply an educated guess based on expert judgment and historic analysis. Ron Jeffries, one of the founders of the agile movement and co-creator of Extreme Programming, has scripted the best way a project manager can respond to that question:

> "Right now, this appears to be a 200-point project. Based on our performance on other projects (or a random guess), with N programmers on it, and your intimate involvement in the project, a project of this size will take between 4 and 6 months. However, we will be shipping software to you every two weeks, and we'll be ticking off these feature stories to your satisfaction. The good news is that if you're not satisfied, you can stop. The better news is that if you become satisfied before all the features are done, you can stop. The bad news is that you need to work with us to make it clear just what your satisfaction means. The best news is that whenever there are enough features working to make the program useful, you can ask us to prepare it for deployment, and we'll do that. As we go forward, we'll all see how fast we're progressing, and our estimate of the time needed will improve. In

every case, you'll see what is going on, you'll see concrete evidence of useful software running the tests that you specify, and you'll know everything as soon as I know it." (Ron Jeffries, as quoted in Mike Cohn's book *Agile Estimating and Planning*)

In agile software development, scope planning is done as part of release planning. Scope definition and many of the practices defined in the PMBOK knowledge area of Project Time Management are done as part of iteration planning, where features are elaborated, tasked out, estimated and assigned. Here the key difference is that planning and design work are done only for the pieces that are being readied to code, and not on the entire system. Scope verification is accomplished within the iteration, as product owners/customers get to review, test, and accept the implemented features. And scope change control is handled by the management of the product backlog, as discussed in the previous section.

Quality Management

Another group of individuals forced to change their mindset is the quality assurance and control staff. Used to being the caboose on a runaway train, they have come to expect shortened time frames for testing, specs that don't match the product delivered, and little interest in their input until the last few weeks of the project. Agile software development brings QA back into the analysis and design of the product, keeping them heavily involved in decision-making throughout the entire life cycle. Because incremental code is being developed each iteration, QA is now testing at the very beginning of the project instead of waiting for something to be "thrown over the wall" at the end. And because throughput is increased, QA is finding itself with a need to become more technical as they must automate testing at all levels, and not just via macros run on the GUI.

In fact, in the nirvana of agile approaches, QA drives the entire software development process, in something called "Test-Driven Development." In this approach, tests are written and automated first, before any functional code is written. Once all the test cases and unit tests are complete, then coding begins – and ends when all of the test cases pass (hopefully by the end of the iteration). This is the ultimate in quality planning, quality assurance and quality control.

Typically, quality planning demands that a plan should be defined that covers how to perform quality assurance and quality control activities as they relate to standard QA practices and to organization policies and procedures. Agile QA team members should still address this need, and work with the project team to determine what tools and technology they will use in writing, running, and reporting tests and results. It is important to engage developers in this definition, as they will be contributing to testing by writing unit tests and helping with the framework for automating regression and acceptance testing. Product owners must also be involved, as they should be helping to define and run the acceptance tests. In agile practices, everyone contributes to defining, maintaining, and enhancing the quality of the product.

Quality assurance, with its focus on preventing defects, is translated into the agile practice of having committed QA resources on the development team that participate in decision-making on a daily basis, throughout the life cycle of the project. Their input during elaboration and design helps developers write better code. More "what-if" scenarios are considered and planned for, as the collaboration between coders and testers gives the coders more insights than if they were to have planned the work on their own. Likewise, the testers gain added insight into the expected functionality from the coders and the product owner, and are able to write more effective test cases for the product.

Quality control places its emphasis on finding defects that have already slipped into the system, and working with developers to eliminate those defects. This bug-checking is done within the iteration, using such techniques such as daily builds and smoke tests, automated regression testing, unit testing, functional and exploratory testing, and acceptance testing. Everyone participates – no one is exempt from the tasks of ensuring that the feature coded meets the customer's expectations.

Human Resources Management

Once staff acquisition has been accomplished, PMBOK notes the processes of Organizational Planning and Team Development. The PMBOK defines organization planning as the process where roles and responsibilities are assigned; the PMBOK defines team development as the development of the appropriate

skill sets and competencies of each individual team member (PMI, 2000). The agile approach to this is to establish cross-functional teams, and allow them to self-organize.

Establishing a cross-functional team means that the development team is no longer made up solely of programmers. Instead, the agile development team consists of all the key players needed to create an increment of working code: testers, analysts, architects, technical writers, subject matter experts, the customer/product owner, and the team lead (project manager, XP coach, ScrumMaster). These individuals bring with them a particular skill set, but as the team matures, each individual will learn more about others' tasks and efforts, and will gradually become more willing and able to pitch in and help in areas outside the individual's normal expertise. Having a cross-functional team eventually comes to mean that you have a group of individuals so committed to delivering the promised features that they will chip in and help do whatever needs to be done in order to successfully finish the iteration.

The team becomes self-organizing as a result of the regular check-ins on the product and process, where the team collaboratively examines what they've accomplished and makes decisions about how they want to continue. This total ownership by the team of the planning, execution, and review of both the product and the process leads groups to a high level of self-directed performance maintained by continuous reflection and improvement.

This is another hard concept for project managers to envision, one that often leads to increased nervousness as project managers begin to wonder what their role will be if they're no longer directing a team's efforts. Here is where the shift from being a command-and-control manager to what Robert Greenleaf described in the 1980s as a "servant leader" must occur (Greenleaf, 1998). Team dynamics don't change overnight, and teams new to agile methods need strong leadership to help them learn how to make group decisions. Becoming a servant leader involves learning how to foster collaboration and reflection as a means to allow your team to do the best work they possibly can. Ongoing servant leadership in mature teams includes responsibilities such as mentoring, guidance, facilitation, and roadblock removal.

Making the Change

Project managers who are willing to make the change to servant leadership will find the journey exciting and rewarding. Watching teams learn how to work together, recognize and be proud of their achievements, and take daily pleasure in their efforts is an incredible thing to witness and be a part of. All it takes is the ability to hand control over to the team, and ask the right questions at the right time in order to help guide them. All the other skills needed – understanding and reminding the team of the process and of the decisions made, scheduling meetings, sharing information, and providing the tools and environment conducive to great work are all part of the project manager's responsibilities today.

The first thing the agile project manager must do is learn the process. Whether you're doing Scrum, Extreme Programming, or another style of agile software development, being the knowledge repository of the methodology's principles and practices is key. The team will look to you for understanding, and will question you on what to do in a particular instance, or ask why a thing is done a certain way. Being able to listen carefully to their issues and concerns, and respond thoughtfully, will help the team learn the new process and accept the disciplines employed by that process. Reading books and articles on agile methodologies is a must; obtaining training by seasoned agile professionals should be considered. One methodology, Scrum, offers formal training and certification. Start your readings by heading to the sites sponsored by the Agile Alliance:

- http://www.agilemanifesto.org/ where you can read about how the movement's original founders came to embrace this new approach to software development, and
- http://www.agilealliance.com/ where you can read through a vast library of articles and stay up-to-date on current trends

Additional references and web sites can be found in the Additional Readings section of this paper.

Once a project manager understands the new process and its underlying philosophy, it is time to learn about servant leadership. Read Robert Greenleaf's books on servant leadership to gain knowledge of how you can make sure your team's needs come first, how to foster the personal growth of your team members, and how you can give up control and allow your team the freedom to do the best work they possibly can.

Finally, becoming skilled at how to properly facilitate team discussions is both an art and a talent that can be learned – and it's a talent that is often vastly underestimated in ensuring the success of a team. A facilitator is a guiding force in helping teams share information, hear that information clearly, and have a voice in expressing opinions, defining options, and reaching consensus. Most people, unaware of the intricacies of good facilitation, believe that they already know how to facilitate meetings – but their style is more that of a meeting director, and often prevents teams from achieving cohesion and fully participating in collaborative decision-making. Project managers wanting to help their teams work well together should start by visiting the International Association of Facilitators website: http://www.iaf-world.org/. Here you'll find trainings being offered, news about facilitation, and a directory of certified professional facilitators that can show you, through example, how to properly facilitate team discussions and planning meetings. As a handbook, you'll want to pick up Jean Tabaka's *Collaboration Explained*, an excellent book on the how's, why's, and what's of good facilitation.

Traditional project managers can make the change to agile project management in much the same fashion as when they first learned project management and the plan-driven approach: self-education, experience, training, and outside assistance. Educate yourself by reading and collaborating with others. Attend training. Obtain certifications. Bring in consultants to ease the transition. And don't forget to involve your team in the journey as well!

A Special Note for Program Management Offices

Higher up the project management ladder lies the PMO, or Program Management Office. This group of managers is concerned with managing groups of projects. Some of the PMO responsibilities include project approval, prioritization, and cancellation, resource allocation across projects, best practices knowledge sharing, and executive reporting on project and resource status. Large corporations that adopt agile processes have to address changes in how the PMO will function.

The business purpose for establishing a PMO remains the same; but the activities performed by the PMO staff are adjusted to better fit in an environment that welcomes change. Because of the low formality of agile practices, the PMO will have to rethink what constitutes best practices, and whether or not a best practice for one team is translatable to another team. Often team dynamics differ so that each team will experience different results when executing practices, and teams should therefore be allowed the freedom to self-organize. However this can be a delicate balancing act, as teams are still required to stay in compliance with overall corporate policies. PMO staff will find themselves serving as liaisons between divisions, as they negotiate minimal process steps for the teams who may still have to provide documentation to support capitalization rules, Sarbanes-Oxley requirements, and other regulatory policies that must be followed.

Gathering metrics may have to be completely redefined, and teams of project managers should work with the PMO to determine what will help both the teams and the executive management in tracking performance and reliability. Project status, metrics, and other reports that normally rolled up in a specific format or tool will have to be revisited, so as not to unfairly burden the development team and yet still provide useful standards of information to stakeholders.

Learning about what changes need to be made at the enterprise level and working to make those changes will also be important. Organization and division issues such as matrixing vs. projectized reporting hierarchies, new ways to reward high-performing teams instead of individual stars, and resource allocation and skill set gaps must be addressed. Common problems are a shortage of QA and product owner personnel, a need for additional technical and soft-skills training, and the time-slicing of individuals who find themselves stretched across multiple projects.

Ongoing training needs must also be addressed. As the traditional “process police,” PMO staff must now morph into “keepers of the process.” The PMO must prepare regular agile methodology training programs for new employees, as well as provide continuing education opportunities to existing staff. New career paths may be defined, and new policies for rating performance should be carefully considered.

Moving from traditional waterfall to agile practices takes time, and project managers at all levels should work to prepare themselves and others to make the transition. In fact, making the transition should be viewed in much the same way as an agile software development project. There will be a backlog of items to be addressed, and management should work together as a team to identify the priority of these items and then go about implementing the changes in an iterative and incremental fashion!

Bibliography

Anderson, David J. Agile Management for Software Engineering: Applying the Theory of Constraints for Business Results. Upper Saddle River, N.J.: Prentice Hall PTR, 2004.

Cohn, Mike. Agile Estimating and Planning. Upper Saddle River, N.J.: Prentice Hall PTR, 2006.

Greenleaf, Robert. The Power of Servant Leadership. San Francisco: Berrett-Koehler Publishers, 1998.

Project Management Institute. A Guide to the Project Management Body of Knowledge. Newtown Square, PA: PMI Publications, 2000.

Additional Readings and Links

Books:

Beck, Kent. Extreme Programming Explained: Embrace Change. Boston: Addison-Wesley, 2000.

Bennis, Warren and Biederman, Patricia Ward. Organizing Genius. Boston: Addison-Wesley, 1997.

Cohn, Mike. Agile Estimating and Planning. Boston: Addison-Wesley, 2006.

Collins, Jim. Good to Great. New York: HarpersCollins Publishers, 2001.

Highsmith, Jim. Agile Project Management: Creating Innovative Products. Boston: Pearson Education, 2004.

Poppendieck, Mary and Poppendieck,Tom. Lean Software Development. Addison-Wesley, 2003.

Schwaber. Ken. Agile Project Management with Scrum. Redmond, WA: Microsoft Press, 2004.

Tabaka, Jean. Collaboration Explained. Boston: Addison-Wesley, 2006.

Links:

The Agile Alliance: http://www.agilealliance.com/

The Agile Manifesto: http://www.agilemanifesto.org/

The Agile Project Leadership Network: http://www.apln.org/

The Declaration of Interdependence: http://pmdoi.com/

International Association of Facilitators: http://www.iaf-world.org/

Jim Highsmith free webinar on agile project management:

http://www.rallydev.com/register_for_webinar_series.jsp#jimPresentation

Rally Software knowledge page: http://www.rallydev.com/agile_knowledge.jsp

Discussion groups:

http://finance.groups.yahoo.com/group/agileprojectmanagement/

http://groups.yahoo.com/group/scrumdevelopment/

http://groups.yahoo.com/group/extremeprogramming/

Agile Project Management (AgilePM, which is derived from DSDM) comes from Agile Business Consortium. Itâ€™s a method by which different, possibly permanent agile teams and non-agile teams are coordinated for the duration of a project. The AgilePM method for managing agile projects consists of a framework, comprising a philosophy, derived principles and four building blocks people, processes, products and applications.Â The Digital Project Manager is the home of digital project management inspiration, how-to guides, tips, tricks, tools, funnies, training, and jobs. We provide project management guidance for the digital wild west where crazy clients, tiny budgets and stupid deadlines reign supreme. Find Out More.