PROGRAM MANAGEMENT FOR AGILE
Retooling Your Approach for Agile Success

Executive Summary
PROGRAM MANAGEMENT FOR AGILE

Retooling Your Approach for Agile Success
The Performance Institute is a private, nonpartisan think tank improving government results through the principles of performance, transparency, and accountability.

This report was sponsored by Robbins-Gioia Inc. (RG), a recognized leader in program management services. RG is dedicated to helping organizations optimize business processes, accelerate change, and establish lasting quality improvements. The authors thank the contributors to the report, specifically Sanjiv Augustine, Diana Berardocco, Ron Bohlin, Jerry Daniels, Doreen Evans, Daniel Furfaro, James Hannon, Bob Johnson, Russell Pannone, Grace Robbins, George Schrader, Rob Thomas, Kelly Sharon, Steven Slater, Jim Chi-Wen Yu, and Kate Zevnik.


Copyright 2013 by The Performance Institute

No part of this book may be reproduced by any mechanical, photographic, or electronic process, or in the form of a phonographic recording, nor may it be stored in a retrieval system, transmitted, or otherwise copied for public or private use, without written permission from the publisher, except for the purposes of official use by the U.S. government.

Printed in the United States of America
EXECUTIVE SUMMARY AND BASIS OF RESEARCH

The Performance Institute undertook this study to identify the role of program management and the program management office in an organizational environment adopting Agile development. Agile has reached a tipping point. A majority of private and public organizations have turned to a modular, or incremental development approach, with Agile being among the most common frameworks. The challenge for program managers and other senior leaders lies in the iterative and incremental approach to requirements and solutions, which only evolve through continuous collaboration with customers and stakeholders. Coupled with a delegation and a reliance on empowered teams to make decisions, new management and control mechanisms appear to be needed as Agile processes are broadly put to use.

To help uncover appropriate practices in an Agile environment, first we set out to learn if program management for Agile is in fact different than traditional program management and Agile project management.1 We also wanted to discover leading practices that exist for managing large, complex programs and the implications to an organization for implementation and sustainment whose projects are developed using an Agile approach.

This study includes a survey of respondents to gauge use of Agile for large program development, case studies, a tools review, and methodology review.

Based on their extensive experience, the authors categorized their findings into eight management practices, which provide insights into how to better manage programs in an Agile environment.

KEY FINDINGS

We based our understanding on the rich resources of data from successful and failed programs that used Agile. The research led us to conclude the following:

- To realize the value of Agile, organizations must use Agile consistently across the enterprise where there may be hundreds of Agile teams working simultaneously;
- There is an overarching vision and organizing function for master planning, resource management, and other familiar program office functions that tie individual projects to organization, strategy, and priorities; and
- Traditional program management techniques are not always aligned with Agile and, therefore, program management for Agile needs to be understood and further defined.

What became apparent to the researchers and authors is that though more organizations are employing Agile, the methods and techniques are implemented in an improvised, unplanned manner, making enterprise implementation difficult, if not impossible. These eight categories of program management practices are suggested to take full advantage of an Agile approach.

1. Evolving vision: Agile projects rely on rapid execution of an overall program in a piece-by-piece, progressively defined fashion. To keep these various pieces from becoming disjointed, overlapping, and dysfunctional, a powerful vision needs to be the guiding force.

2. Architecture: Without a general framework that guides the definition of individual initiatives, it is unlikely that those initiatives will eventually add up to an integrated whole. Without some sense of common structural guidelines, integration of the pieces becomes difficult or impossible. A guiding definition of the enterprise and systems architecture is even more critical in an Agile environment than in a traditional top-down development program.

3. Dynamic team: In an Agile program, many small work teams need to form rapidly, come up to speed quickly, evolve significantly as the work takes shape, disband without disruption, and reform as needed to evolve with the program. A conventional mindset to team formation and building will not be effective.

4. Collaboration: Multiple small teams with fast-moving concepts create a significant potential for disconnects and redundancies that can result in wasted time and energy and destructive internal competition. The need for simple but effective mechanisms to ensure frequent and meaningful cross-team communication and collaboration is enormous.

5. Decision making: Traditional programs move forward in fewer, larger bundles of activity, typically with longer distances between the need for key decisions. Agile programs require a decision-making process that is much more responsive in dealing with a larger volume of smaller incremental decisions.

6. Project management: The first inclination during a transition to Agile is to jettison the bureaucratic, documentation-heavy management practices that are believed to have caused lengthy projects and frustrated customers and developers. A closer look shows that the implementation—not nec-

---

1 Most Agile methods, scrum in particular, do not include a role called “project manager.” Without a specific person tasked with performing all managing duties, those responsibilities are distributed among the other roles on the project, namely the team, the scrum master and the product owner.
necessarily the practices themselves—is to blame. Transitioning to an Agile approach provides the organization with the opportunity to examine and refine project management practices.

7. Implementation strategies: By their very nature, Agile programs are likely to produce end products at significantly different points in time. There is often value in moving pieces to implementation rapidly rather than waiting for “the whole” to be ready to launch by conventional means. This, however, can produce very different needs for roll-out strategies, communications tactics, workforce readiness initiatives, etc. It also produces an environment where some implementation activities begin long before other related development initiatives have matured.

8. Sustainability: Agile programs, by their nature, often skip past rigorous documentation steps that are fundamental to a typical development. But without reasonable attention to documentation, change management, and ongoing governance methods, those systems and processes can rapidly become increasingly difficult to sustain and evolve to meet changing conditions and requirements.

OVERALL FINDING

The summation from the eight management practices includes:

- The ability to harness Agile engineering’s potential lies with the ability to incorporate the Agile philosophy consistently, across operating units.
- Due to organizations’ needs to conduct multi-year planning, tie development to larger programs and strategies, and track performance, program management must play a role to ensure the increments from Agile and self-organized managed teams are developed to overall organizational goals.
- There often exists a gap between applying Agile methods and the knowledge/skill level to implement an Agile approach.

This research report provides greater perspective on these findings coupled with a survey of common Agile methods and tools.
# TABLE OF CONTENTS

<table>
<thead>
<tr>
<th>Page</th>
<th>Section Title</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>9</td>
<td>PROGRAM MANAGEMENT FOR AGILE: PERSPECTIVES</td>
<td>The Perspectives chapter is an in-depth discussion of eight key areas the program management office must develop to realize the benefits of an Agile approach applied to large, complex, and risky programs. In addition, the PMO must be able to integrate smaller, stand-alone projects for enterprise performance management.</td>
</tr>
<tr>
<td>27</td>
<td>AGILE SURVEY RESULTS AND ANALYSIS</td>
<td>In late 2012, Robbins-Gioia, LLC conducted a research study on the maturity of management practices for Agile programs in organizations. The results indicate a high level of interest and experimentation with Agile development, but relatively low levels of changes to the PMO practices.</td>
</tr>
<tr>
<td>31</td>
<td>SUCCESSFUL APPLICATION OF AGILE: CASE EXAMPLES</td>
<td>This section contains case studies on organizations that have begun to apply Agile to large programs or across the enterprise. They show the implications for managing enterprise programs with Agile and lessons learned.</td>
</tr>
<tr>
<td>39</td>
<td>AGILE METHOD AND MODEL REVIEW</td>
<td>In addition to a review of the more popular Agile methods such as scrum and XP, this section reports on how some organizations have begun to develop policies and processes for incorporating Agile or an incremental approach.</td>
</tr>
<tr>
<td>51</td>
<td>AGILE MANAGEMENT TOOL REVIEW</td>
<td>This section provides an overview of popular tools used to manage in an Agile environment.</td>
</tr>
<tr>
<td>54</td>
<td>TOOL SNAPSHOTS</td>
<td>This chapter provides detail on the most common Agile tools for program management.</td>
</tr>
<tr>
<td>63</td>
<td>APPENDIX A: AGILE GLOSSARY OF TERMS</td>
<td></td>
</tr>
<tr>
<td>69</td>
<td>APPENDIX B: FURTHER READING</td>
<td>An annotated bibliography for the sources of the independent research and continued reading.</td>
</tr>
</tbody>
</table>
The overall vision for the project is established early with the sponsor/product owner, along with a cost estimate and confidence range. But the allocation of the budget to the increments is delayed until the increment planning. With the active participation of the customer in priority setting, the increment development team builds only what is needed, and in priority order—and stops only when the time/money is depleted.

1 Queuing theory is widely credited to Agner Krarup Erlang (circa 1909) at the Copenhagen Telephone Company, who wrote a paper applying probability and statistics to telephone switching problems. Erlang was able to estimate the probability that a call would be blocked at various levels of capacity utilization in the queue. He discovered that capacity utilization increased queues exponentially and that variability of the items in the queue increased the queues linearly.


By working off the "feature backlog" it is possible to hold project costs to the agreed "should cost" estimates, because usable functions have been developed and implemented. Stopping development still results in a percentage of the project being implemented.

Keeping the vision cohesive with stakeholders as iterations are defined and implemented becomes the driving principle for the project leaders, and replaces the dogma of scope control. Translating vision into capability is a requirements process, and how requirements are managed in an Agile environment at the macro-project, release, and increment levels, is at least equally important for the Agile program manager as the traditional PM.

In an interview, Mark Schwartz,

"This creative design process is complicated by the generally poor status of most requirements descriptions. This is not because the users or the system’s designers are incompetent but because of what I call the requirements uncertainty principle. For a new software system, the requirements will not be completely known until after the users have used it."

The true role of design is thus to create a workable solution to an ill-defined problem. While there is no procedural way to address this task, it is important to establish a rigorous and explicit design process that can identify and help to resolve the requirements uncertainties as early in the developmental process as possible."

This requirements uncertainty principle is generally seen as a cornerstone concept of Agile development. The requirements uncertainty principle further supports the concepts that knowledge-based projects are best managed and developed incrementally and in small batches. How requirements are translated into functional capability and by whom remains a vital question that organizations need to answer.

The business analyst (BA) role seems neglected by the Agile scrum methodology. To ensure close collaboration between the team and product owner, scrum ensures that the necessary elements are effectively communicated directly to the team without complex documentation. In the Agile environment, the BA works to ensure the increment is adequately documented while balancing the need to move quickly and ensure continuity of information. It is essential that functional documentation is adequate and representative of the solution to be developed and delivered.

As such, the business analyst’s input is invaluable to the product owner. The business analyst may be called upon to play the role of product owner during the daily interplay of understanding customer uses; he/she then becomes responsible for core components of the product.

In Agile, development is done by small teams empowered to make trade-off decisions in order to keep the project moving forward quickly. In large organizations with many stakeholders who have varied interests, trade-off decisions made quickly within an iteration can have unknown consequences to other parts of the organization. Traditional PM approaches mass coordination often with heavy documentation and gatekeeping.

**MINI CASE**

A municipal government ERP project had grown out of control. Root cause analysis indicated that the project team had lost track of the vision. The city council had authorized bonds to cover the ERP’s cost because it needed (1) more accurate and timely data to fulfill fiduciary responsibilities and (2) the opportunity to improve revenue by better accounting. Instead, the project team developed a 700-page specification that obscured sponsor expectations.
reviews to minimize risk to the enterprise. Replacing documentation in the Agile environment requires collaboration. Besides including stakeholders on the iteration team, some organizations have begun adjusting the release reviews explicitly to include the trade-off decisions (cost versus time versus functionality) fully vesting the customer in the decision. The teams are empowered to make tentative trade space decisions within the parameters established in early discovery sessions that establish a shared vision for the ultimate program outcome.

The purpose of the vision is to give self-organizing (autonomous choice of action) teams proper direction. The vision describes the what, who, why, when, how, and constraints and assumptions for the product (solution) to be developed and delivered as described below.

WHAT:
- Summarizes the major benefits and features the product will provide.
- Gives context to readers, especially those in the product development and delivery team, by defining the business context for the product.

WHO:
- There are a number of stakeholders with an interest in the development and not all of them are end users. Think of this as outside-in design.
- Stakeholders will have different views of what a “requirement” is and provide the background and justification for why the requirements are needed.

WHY:
- Functional description in context of organization strategy and mission.
- Need, opportunity, and benefit.

WHEN:
- Begins the process of project scheduling by clarifying the stakeholders’ time expectations regarding the product.

HOW:
- Describes at a high level the product (solution) development approach.
- Begins the activity of system engineering.
- High-level design.
- High-level architecture.

CONSTRAINTS AND ASSUMPTIONS:
- Constraints are restrictions that affect the actual adoption of the change. The most typical constraint is the budget, which limits the scope, staffing, and schedule.
- Assumptions are factors considered to be true, real, or certain and involve a degree of risk (e.g., if a change requires new skills, the assumption is that people will receive the necessary training).

A broad vision statement describes outcomes for the project with a set of features described as a backlog. Once an iteration has been completed, the overall plans can be revised to reflect the completed work, gained knowledge about the customer needs and opportunities, and development practices, remaining priorities, and budget and schedule status.

Initial cost and schedule estimates can be revised to reflect the actual cost and timing of the completed work. In contrast, with traditional waterfall project management, this analysis is documented in detail at the beginning of the project for the entire scope of work.

In a discovery session, the vision is described in a few short pages with teamwork keeping three primary stakeholders (product owner, scrum master, and subject matter experts) together, rather than a significant effort devoted to documenting strategies, project plans, cost and schedule estimates, and requirements for a full system. The discovery session is not a part of established methods, such as scrum, but is developed as a matter of necessity on large programs (see the case study on Agile and a large government agency in this report).

The next issue to be resolved is business process integration. Individuals and independent teams tend to do similar tasks in different ways. Agile requires that

**MINI CASE**

The Department of Defense guidance on incremental development, Business Capability Life Cycle’s (BCL) use of a business case (which integrates program-level plans and information for decision makers) and program charter (which outlines roles, responsibilities, and organizational agreements) reduces the amount of documentation that must be coordinated, particularly at the Office of the Secretary of Defense (OSD) level, trimming time and cost. Program-level documentation may still be coordinated and approved within the component, but does not need to be approved by OSD. This approach places focus on the need, solution, and risk—not the amount of documentation.
subject matter experts and business team members have the required knowledge. This practice stresses that staff involved in fast-paced iterations must be experts in the processes being automated, and in those iterations, able to reduce delays. For example, a team member representing financial customers must be fully familiar with the needs of those customers—while realizing every branch and division works a little differently, often for valid local contextual reasons. There is a dynamic tension between the need to align standardized business processes and for flexibility.

2. THE NEED FOR A SOLID OVERALL ARCHITECTURE TO GUIDE DIRECTION

Without a general framework guiding the definition of individual initiatives, it would be very unlikely that those initiatives would add up over time to an integrated whole. Without some sense of common structural guidelines, integration of the pieces becomes difficult or impossible. A guiding definition of the enterprise and systems architecture is even more critical in an Agile environment than in a traditional top-down development program. Should the PMO be more integrally involved in the development and evolution of those architectures as a fundamental element of the program? What incremental skills and methods does that require?

Under an Agile approach, as with any methodology, if project teams work under the assumption that they can do anything and use any technology they want, chaos typically results. Functionality and information will be duplicated and reuse will occur sporadically, if at all. Systems will not integrate well; in fact, they will conflict with one another and cause each to fail. Costs will skyrocket because similar products from different vendors, or even simply different versions of the same product, will be purchased and used within production. Although each individual project may be successful, as a portfolio there may be serious challenges—but this doesn’t have to happen.

Applications must co-exist effectively with the other systems within an organization. Therefore an application must be minimally developed so that it doesn’t cause adverse effects on other systems. Ideally an application should be built to take advantage of and enhance a shared infrastructure. Each system must be built so it can fit into an existing environment—better still, so that it reflects the future vision for an organization. This sort of information should be captured in an enterprise architecture (EA) in both current and future state models. Agile enterprise architects must ensure this happens in an effective manner; ensure the needs of the business stakeholders are understood and anticipated; and support project teams in their development efforts. It is critical that the Agile EA process be:

- Business driven. The architecture effort should be owned and driven by an organization’s business, not by the IT department. In many organizations the IT department initiates EA programs, typically because the business doesn’t know to do so—but IT must educate the business on its need to own EA.

- Collaborative. The best enterprise architects, just like the best application architects, work closely with the intended audience of their work, both on the business side of the house as well as the IT side. These enterprise architects will roll up their sleeves and become active members of development teams, often in the role of architecture owner on Agile teams, or as an architect on more traditional teams. Their mission is to ensure that the development teams work to leverage the EA, mentor developers in architecture skills, and identify what works and what doesn’t in practice, so that they can evolve the EA accordingly. Enterprise architects who don’t participate actively on development teams (merely holding architecture reviews isn’t active participation) run the risk of being removed from the program and thus easy to ignore.

- Focused on producing valuable artifacts. The most valuable artifacts useful to the intended audience are lightweight and executable. Many EA programs run aground when the enterprise architects focus on artifacts. A detailed enterprise data model indicating suggested data attributes would be intellectually interesting to develop, but a list of legacy data sources with a high-level description of their contents would be immediately valuable to many development teams in their development efforts.

MINI CASE

The U.S. Marines Corps, by its mission, plans and fights in a distributed manner while the Army plans centrally, again defined by its mission. The impact on communications and connectivity requirements as well as taxonomy (e.g., distance measured in miles versus kilometers) can cause very different decisions! A system that supports the Army and Marines must have a vision for both contexts.

---

teams. A detailed model depicting desired web services would be useful, but an actual collection of working services that can be reused immediately would be best.

- An explicit part of development. Part of the architectural advice is that delivery teams should work closely with their organization’s enterprise architects so they can leverage and at times help to build out the common infrastructure. Disciplined Agile teams realize that they can benefit greatly by doing so.9

There is widespread consensus that a high-level or top-down view 5-10 or more years out is required to contain duplication and costs and provide a framework for iteration developers. Bottom-up architecture addresses questions coming out of projects right now—questions that should be answered quickly. Aligning top-down and bottom-up architectures can improve the ability to quickly make project-level decisions that support the long-range vision. Agile has the effect of forcing architects into conversations instead of taking another six or nine months to finish the next version of the enterprise architecture. A common directive from senior leaders is to work with the development teams to decide collaboratively which architecture views are needed—shifting from audit and compliance to facilitator.

The Defense Information Systems Agency’s (DISA) framework shown in figure 1 reflects how this multi-level planning might be carried out. While not in this framework, one can imagine a fourth high level that might be called the line of business or mission capability roadmap. Architecture has had a hard time finding a home in most organizations. Technology groups may depend on the architecture to support modernization. However, as stated above, EA must be owned and driven by the business. An emerging practice is to relook at the enterprise PMO (EPMO) and make architecture a function within it.

This model uses the principle of tiered accountability by assigning responsibilities to the lowest appropriate and permissible statutory and regulatory level. This approach strengthens accountability, reduces bureaucracy, and accelerates positive outcomes. In addition, the functional sponsor, who represents the users and champions the needed capability throughout, plays a critical role in delivering successful capability. The functional sponsor works with the program manager (PM) during the entire process, and while the PM is responsible for the materiel portions of the capability, the functional sponsor is responsible for the remaining doctrine, organization, training, leadership and education, personnel, facilities, and policy portions of the solution; justifying the program; securing funding; and eventually ensuring the solution has met the need that the users originally identified.

---


3. THE NEED FOR A DYNAMIC APPROACH TO TEAM STRUCTURES

In an Agile program, many small work teams need to form quickly, come up to speed quickly, evolve significantly as the work takes shape, disband without disruption, and reform as needed to evolve with the program. A conventional mindset to team formation and team-building will not be effective. The PMO needs to be much more proactive and provide a source of needed flexibility to the organization to maintain energy, pace, and productivity, and retain and effectively redeploy skills to meet evolving needs. This process implies the need for some different human capital management capabilities on the part of the PMO, particularly in regard to staffing.

Staffing considerations for an Agile team includes filling a broad-cross section of skills spanning several organizational lines of control—much like a traditional integrated process team (IPT). The need for cross-functional communication doesn’t change with an Agile approach. Instead the communications are much more dynamic, focused, and based on daily rather than monthly interactions. The difference for Agile projects is the short time frame, dedicated attention to the iteration, and close working conditions in the Agile environment. When staffing for the iteration the project manager considers:

- Team composition based on the product backlog
- Co-location of teams or investing in online meeting and collaboration tools

Despite the fear of Agile methods leading to (or embracing) anarchy in the engineering groups, Agile demands more standardization across the organization than traditional methods. With programs that last 12 to 18 months or longer, the PM has the ability to recruit and train team members and even develop program-specific processes and procedures. The Agile PM working in short increments does not have that luxury. The Agile team has to quickly form and immediately start working on the backlog (or whatever requirement process is used). The only way this can happen is for a common method across the organization supported by HR practices that facilitate a free flow of skills in and out of projects.

A project-based organization such as this that can spontaneously form teams to work opportunities or needs and then disband is the vision established by the PMI® Organizational Project Management Maturity Model or OPM3© shown in figure 2.
Business processes are the collection of activities that create value. According to strategy authority Michael Porter, these activities are the ultimate source of competitive advantage. “There are just two types of competitive advantage. Cost advantage arises because a firm can cumulatively perform the discrete activities in a business more efficiently. Differentiation depends instead on a firm’s ability to perform particular activities in unique ways that create buyer value.”

Level five of the OPM3 model envisions teams coming together to solve problems and then disbanding to reform around another issue elsewhere in the organization. Rather than independence, this requires a highly developed sense of common procedures for planning, managing, and performing work on projects, as well as cultural norms for working across organization boundaries.

Business and HR practices that change to support dynamic team creation include:

- Performance reviews that encourage essentially multi-matrix management by soliciting and integrating performance from multiple sources and annual goals based on achievement of team and organizational objectives in addition to organizational or unit goals.

---

• Talent or resource management tools and practices that support the identification of competency areas and track demand and supply for key skills and knowledge

Steps to get there are:

• Rewrite roles and responsibilities of team members
• Consider the use of an Agile coach on the team
• Pre-award training of new team members when the effort is dependent on procurement actions
• Master resource planning to identify and remediate bottlenecks
• Consider hiring an independent (outside) change agent and coach

Potential modifications to the team makeup are:

• Embed a compliance, information assurance, operational test, and/or privacy representative on the team rather than sending the product for review and approval at the end of the iteration.
• Test early and often throughout the life cycle. Testing during instead of after software code delivery reduces risk and remediation costs. Evaluations can occur at the project and organizational level. For example, at the project level, the iteration is reviewed at its completion in a retrospective. At the organizational level, processes are reviewed for opportunities to improve the approach.
• Obtain stakeholder/customer feedback frequently and closely. For example, feedback is obtained during the iteration and at its completion at a retrospective. This practice is linked to reducing risk, improving customer commitment, and improving technical staff motivation. Agile software products are presented to customers for approval incrementally at the end of each iteration, instead of presenting complete products for approval at waterfall milestones. However, some customers mistrust iterative solutions. One U.S. government agency official said customers expecting a total solution feared that the initial demonstrations of functionality provided in the current iteration would be considered good enough, and they would not receive further software deliveries implementing the remainder of their requirements. At another agency, an official said this fear contributed to customers finding it difficult to define done. Specifically, customers were challenged in defining when each requirement would be considered done because they were afraid that this would be viewed as meaning all related functions were being met.

Procurement practices may not support Agile projects. Agile projects call for flexibility to add the staff and resources needed for each iteration and adapt to changes from one iteration to the next. Federal procurement practices do not always support the flexibility required. For example, federal contracts that require onerous, waterfall-based artifacts to constantly evaluate contractor performance are not needed in an Agile approach when the contractor is part of the team whose performance is based on the delivery of an iteration. Furthermore, accommodating contractor staff and task changes for different iterations is challenging because contracting officers require at times lengthy structured tasks and performance checks.

Compliance reviews can be difficult to execute within an iteration. Iterations may incorporate compliance reviews to ensure, for example, that agency legal and policy requirements are being met. One federal agency official reported a challenge obtaining compliance reviews within the short, fixed time frame of an iteration because reviewers followed a slower waterfall schedule. Specifically, the official said that compliance reviewers queued requests as they arose and that the reviews could take months to perform. This caused delays for iterations that needed such reviews within the few weeks of the actual work.

The Defense Information Systems Administration (DISA) approach to forming teams as depicted in figure 3 pulls people from four distinct organizations: the business/functional/operations unit; analysts and developers from OCIO (in this case of IT) and hired on contract; and the independent test group.

Table 1 shows a RACI (responsible, accountable, consulted, informed) matrix or team structure proposed by Russell Pannone. Every organization will want to establish its own RACI matrix to define areas of responsibility, but this table represents a common one. The matrix delineates the roles required of a highly disciplined team.

4. THE NEED FOR INCREASED, EFFECTIVE, AND EFFICIENT COLLABORATION ACROSS TEAMS

Multiple small teams with fast-moving concepts and tasks create significant potential for disconnects and redundancies that can result in waste and destructive internal competition. The need for simple but effective mechanisms to ensure frequent and meaningful cross-team communication and collaboration is enormous. That collaboration can be facilitated by
the use of technology to provide an effective platform, which needs to be purpose-driven, valued by team members, easily adopted and minimally intrusive into the focus and operation of the teams. The PMO is the logical and appropriate entity to define those mechanisms and create, host, and facilitate the use of a platform for collaboration. Although this type of support is very different, it is analogous to that provided by a typical PM schedule monitoring tool.

Agile development emphasizes collaboration more than traditional approaches do. For example, to coordinate the many disciplines—such as design and testing—involved in an iteration, customers work frequently and closely with technical staff. Furthermore, teams are often self-directed, meaning tasks and due dates are managed within the team and coordinated with project sponsors and stakeholders as needed to complete the tasks. In contrast, in traditional project management, customers and technical staff typically work separately, and project tasks are prescribed and monitored by a project manager, who reports to entities such as a program management office.

Key Agile collaboration and communication practices center around the very short but frequent team meetings (e.g., the daily scrum). During the meeting, each team member answers the following three questions:

1. What have I accomplished since the last daily scrum?
2. What am I going to do before the next daily scrum?
3. What obstacles are in my way?

Daily scrums improve communications, eliminate other meetings, identify and remove impediments to development, highlight and promote quick decision making, and improve everyone’s project knowledge. Combined with graphic and visual progress tracking systems, the daily team meetings are used to manage work, and in a team setting, status the progress of planned activities and product elements. In many environments the daily stand-ups relieve the PM of maintaining detailed project plans and reporting “weekly” progress. Plans are maintained at a high level and progress is reported by customer acceptance of increments.

Additional management practices include:

- Tracking progress daily and visibly. For example, a progress chart is posted openly in the team’s workspace, with daily or near daily revisions to
reflect ongoing status. The highly transparent visibility into small pieces of work is very different from the traditional practice of using a software application to reflect the status, which is updated weekly or monthly and shared through a dashboard and/or static reporting.

• Collaboration is essential, but change is hard. Posting project status on team room wall charts is initially viewed as intrusive. Technical team members may prefer to work autonomously. Development teams may not like showing work constantly to customers, and customers may not want to see incremental product variations. Successful transitions acknowledge these concerns while firmly using open communications and work environments. They plan extra time to support change management initiatives and prepare for an initial decrease in productivity.

• Requiring frequent input from staff. This activity takes more time than that required for waterfall development projects, which allows more sporadic participation. For example, two officials said their agencies were challenged in dedicating staff with multiple, concurrent duties to Agile teams because staff could not be spared from their other duties. The third official said stakeholder commitment is challenging to maintain when stakeholders rotate frequently and new staff need to learn the roles and responsibilities of those being replaced.

5. THE NEED TO ADJUST PROJECT, PROGRAM, AND BUSINESS ANALYSIS MANAGEMENT PRACTICES

The first inclination during a transition to Agile is to jettison the bureaucratic, documentation-heavy management practices that are believed to have caused lengthy projects and frustrated customers and developers. A closer look shows that the implementation—not necessarily the practices themselves—are to blame. Transitioning to an Agile approach provides the organization with the opportunity to examine and refine program, project, and business analysis (BA) practices. For example, the traditional role of the business analyst involves working with the project owner and stakeholders to create a single requirements database and document a set of use cases at the beginning of the project. In an Agile project, the business analyst creates a continuous requirements management approach where the customer’s future needs and opportunities are elicited and communicated to the rest of the development team. A BA working as part of the product owner team facilitates interaction with the development teams depending on the stage of the
iteration. Early on, the BA supports understanding of features. Subsequently the BA supports interpretation of customer needs during development, all while looking two or three iterations ahead. The core of these requirements needs to be maintained and fed back into the vision.

CONTRACTING

Before an organization attempts to roll out Agile methods, it is wise to see if first it can meet contracting requirements. Administratively, because it requires adherence to contracting regulations and product development standards, the federal government is a special case. Most software to be developed for military use, for example, requires different formal documentation and process than are associated with an Agile project. At a minimum to engage in Agile a third-party product developer would need to:

- Obtain an agreement from the contracting officer’s technical representative (COTR) that Agile is an acceptable approach to be used on the project.
- Receive a waiver for each waterfall document required by the applicable standard for product development, but not delivered.
- Submit a tailoring document detailing the product development cycle to be employed, together with an identification of the controlling documentation to be produced.
- Consider indefinite-delivery indefinite quality (IDIQ) contracts, that support a modular, Agile-style product development.
- Consider multiple-award task order contact frameworks to promote contracting for Agile development modules, coupled with competition among a small group of contract holders for each module. (This would require architecture/integration support either through the government or a contractor.)
- Allow task orders to be priced on a time and materials basis.
- Use past performance extensively rather than upfront performance or other requirements in making future task order awards.
- Eliminate requirements in the task orders for Agile projects, instead using objectives.
- Allow a simple, Agile-style change order process.
- Use rolling contracts that support short development iterations. Alternatives to statements of work are performance-based contracts and statements of objectives that focus on outcome and how to get there, as opposed to building a system.

REPORTING

Managers can help their teams develop business outcome-based metrics by focusing on the customer’s definition of value, which in Agile involves quality and progress.

Several common metrics give teams the detail they need without overwhelming them:

- Velocity: The number of features a team can deliver during a sprint is the principal Agile metric, as it allows the team to accurately predict and plan progress, keeping projects on schedule and within budget.

MINI CASE

As one government agency transitioned to Agile, its members experienced culture shock firsthand. The stakeholders were used to working in a serial life cycle where significant communication occurred through documentation. In an effort to build trust and encourage collaboration, program leaders:

- Built a collaborative, open work space by clearing cubicles from a large section of the office and moving the team into an open environment
- Encouraged stakeholders to dedicate their team representative to the sprints and co-locate with the development team
- Set the communications ethic and trained the workforce in handling information overload
- Fostered an environment of tolerance for different ideas and opinions, but not for personal slights or drama-infused conflicts
- Facilitated regular face-time between knowledge workers
- Reallocated context tasks so knowledge workers could focus on high-value activities
- Managed the workforce based on results achieved rather than by time at work

MINI CASE

For the Department of Veterans Affairs, FY 2011 marked the first full year in which all IT projects were managed and tracked through the Project Management Accountability System (PMAS), which demonstrated the successful delivery of 89 percent of all project milestones, compared with the IT project success rates across industry and government at approximately 32 percent. For the 12 months that ended on September 30, 2011, VA managed 101 IT projects, established a total of 237 milestones, and successfully executed 212 of those milestones. The purpose of PMAS is to ensure VA’s IT investments result in successful delivery of functionality that serves veterans. Since 2009, VA attributes at least $200 million in cost avoidance to implementing the PMAS program to manage projects. It also cites PMAS and other strong management disciplines as the main reason that VA has requested no increase from its 2010 IT spending level in its 2011 and 2012 budgets. VA introduced PMAS in mid-2009, and soon announced 45 IT projects would be paused and reformed.


- Burnup/burndown: A burnup chart shows how many features the team has promised to deliver, while a burndown chart shows how many features it has completed.
- Running tested features (RTF): Depicts how many features in each sprint have passed acceptance tests.
- Defect density: For IT or software Agile projects, the number of bugs is a critical quality metric.
- Earned value: communicates a project's progress toward delivering its expected goals on Agile projects. The earned value is based on data such as completed user stories instead of the more traditional use of planned costs of work packages. This measure is similar to velocity and used primarily when EVM reporting is required.

TABLE 2: SUMMARY OF PRACTICE TAILORING

<table>
<thead>
<tr>
<th>PMO Activity</th>
<th>PMO Impact</th>
</tr>
</thead>
<tbody>
<tr>
<td>Burnup/burndown</td>
<td>Reduced feedback loops</td>
</tr>
<tr>
<td>Running tested features</td>
<td>Improved test coverage</td>
</tr>
<tr>
<td>Defect density</td>
<td>Increased bug tracking</td>
</tr>
<tr>
<td>Earned value</td>
<td>Enhanced project visibility</td>
</tr>
</tbody>
</table>

Table 2 is a summary of the practice tailoring that often supports a transition to Agile. Perhaps the most important message is that most of the program and project management practices are needed with an Agile approach, but they need to be adjusted to the organization, project, and team.

6. THE NEED FOR A MORE RAPID INCREMENTAL DECISION-MAKING APPROACH

Traditional programs move forward in fewer, larger bundles of activity, typically with longer distances between the need for key decisions. Agile programs require a decision-making process that is much more responsive in dealing with a much larger volume of smaller incremental decisions. Sometimes smaller decisions are more difficult because they lack a typical context, and have received more limited focus on alternatives. Management behavior (and knowledge) may need to change to ensure that decision making does not become the limiting factor in schedule success, and the sum total of all of these small decisions add up to good overall direction and outcomes. What specifically can the PMO do to facilitate that process?

19 Ibid.
### Table 2. Program Management for Agile: Examples of Appropriate Project Applications

<table>
<thead>
<tr>
<th>MANAGEMENT PRACTICE</th>
<th>AGILE APPROACH</th>
<th>TRADITIONAL DESIGN/BUILD</th>
</tr>
</thead>
<tbody>
<tr>
<td>JUSTIFICATION—BUSINESS CASE &amp; CHARTER</td>
<td>Project value proposition within a governance framework.</td>
<td>Project value proposition within a governance framework.</td>
</tr>
<tr>
<td>SCOPE</td>
<td>High-level feature hierarchy fully established early in project with WBS and allowance for discovery of operational needs and incremental prioritization.</td>
<td>Detailed WBS established early in project and put under configuration control to manage scope creep; customers asked to commit to needed features and functions early in the project life cycle.</td>
</tr>
<tr>
<td>REQUIREMENTS</td>
<td>Vision and macro requirements established at initiation, while detail emerges later as functional and iterative increments. Modules are prioritized with customers in a program roadmap. Detailed requirements discovered and prioritized during incremental development. Documentation only to the level needed to maintain the increment.</td>
<td>High-level vision and robust detail established early in the life cycle before building begins. Strict configuration control and management of requirement changes at all levels. Reliance on written documents to communicate with operations for approval.</td>
</tr>
<tr>
<td>SCHEDULING</td>
<td>Master schedule developed with customer based on a functional release schedule. Details within the increments managed within the increment team and not necessarily migrated to the master schedule.</td>
<td>Rolling wave—activities broken down to fewer than two weeks within the wave. Detailed activities become part of the baseline integrated schedule.</td>
</tr>
<tr>
<td>INTEGRATED PLANNING</td>
<td>Real-time integration planning develops over time, often in weekly sessions with team leaders. Major integration points established during planning focused on “what” and “why” with “how” during incremental module development.</td>
<td>Master schedule with dependencies established early. Budget and schedule tied to the integrated plan. Team members work to the plan and customer notified when the plan must change to meet objectives.</td>
</tr>
<tr>
<td>PROCUREMENT/CONTRACTING</td>
<td>Collaborative decision making among the parties can relieve pressure on liability, warranty, and similar issues. The contract must support relationships based on collaboration, transparency, and trust in the operation of the Agile process and incremental delivery.</td>
<td>A performance work statement or statement of objectives is a best practice typically with a fixed priced contract. The contract process compartmentalizes risk to the parties. Focus on project deliverables.</td>
</tr>
</tbody>
</table>

### SKILLS AND TRAINING

<table>
<thead>
<tr>
<th>SKILLS AND TRAINING</th>
<th>AGILE APPROACH</th>
<th>TRADITIONAL DESIGN/BUILD</th>
</tr>
</thead>
<tbody>
<tr>
<td>COLLABORATION</td>
<td>Daily team integration with weekly management engagement; work across organization’s business silos. Share understanding and discoveries immediately.</td>
<td>Reliance on a good project plan that enables individual, team, and organizational contributions. Status, issues, and risks are managed through project reporting.</td>
</tr>
<tr>
<td>LEAN SIX SIGMA</td>
<td>A systems approach to collaboration with a focus on eliminating non-value added activity during development.</td>
<td>Focus on streaming processes prior to development and during sustainment.</td>
</tr>
<tr>
<td>BUSINESS ANALYSIS</td>
<td>Business process modeling and requirements management through user stories or similar mechanisms.</td>
<td>Business process modeling precedes detailed requirements definition and project planning; intensifying again during implementation.</td>
</tr>
<tr>
<td>STAFFING</td>
<td>Semi-autonomous small teams work collaboratively as self-organizing, self-regulating, colocated groups of individuals. Product owner representative included in the development team. Teams consist of senior staff working as peers.</td>
<td>Large integrated teams work on assigned, predetermined activities. May be collocated or dispersed. Use of a wide mix of skills and experience often relying on a few senior staff to guide junior staff.</td>
</tr>
</tbody>
</table>

### ORGANIZATIONAL

<table>
<thead>
<tr>
<th>ORGANIZATIONAL</th>
<th>AGILE APPROACH</th>
<th>TRADITIONAL DESIGN/BUILD</th>
</tr>
</thead>
<tbody>
<tr>
<td>PRIORITIZATION (CUSTOMER INVOLVEMENT AND FEEDBACK)</td>
<td>Portfolio priorities established collaboratively across the organization. Project priorities can be established just before incremental development with customer.</td>
<td>Portfolio and project priorities established during (typically) annual budget process and reviewed quarterly.</td>
</tr>
<tr>
<td>CUSTOMER INVOLVEMENT</td>
<td>Throughout the project.</td>
<td>Focused on early definition and planning and final acceptance.</td>
</tr>
<tr>
<td>VALUE (ALIGNMENT TO STRATEGY)</td>
<td>By working in small increments with the ability to prioritize within the increment, there is a nearly continuous alignment of project work to value.</td>
<td>A tendency (though not a best practice) to measure performance strictly against the project plan delivery dates. Value realization depends upon full implementation.</td>
</tr>
<tr>
<td>MANAGE CONSTRAINTS</td>
<td>Portfolio management used to identify and reallocate resources incrementally based on continuous identification of priority project components. Mid- and long-term resource management based on strategic alignment versus aggregate project plans.</td>
<td>Constraints and mitigation/buffers are built into the project plan.</td>
</tr>
<tr>
<td>TRANSITION TO SUSTAINMENT</td>
<td>Enterprise view by release. Customer acceptance of increments; collaboration continues throughout the life cycle. Prioritize and organize across organizational silos.</td>
<td>System readiness review with formal handoff to operational business units.</td>
</tr>
</tbody>
</table>
Agile processes assume that plans will change, perhaps significantly during the project. In an uncertain and unpredictable environment empirical methods should be used to monitor projects and direct change. The PMO must support teams to develop and use a more empirical approach while maintaining definitive measures, resource management, and alignment to the organization’s strategy. As Yogi Berra famously said, “It is hard to make predictions, especially about the future.” But a sound plan and ability to monitor progress in near real time enables the team to react to change. The PMO has two roles. First, it must help the increment monitor progress with day-to-day decisions using practices that are consistent across the organization. Consistency is important for the free flow of people and information. Otherwise there is a steep learning curve every time a new project or new team is formed. Senior leaders must be able to quickly assess the status and performance of all projects without needing to delve into the specific developmental frameworks. Second, the PMO needs to package all the increments across the company into products and capabilities that senior leaders can use to make decisions on strategic direction and resource allocations.

The PMO can also help with the paradigm shift on trade-off analysis. Traditional trade-off models try to fix scope (i.e., requirements) by playing off cost or time. With the Agile approach the time and increment teams are fixed; and as the customer has established a benefit to cost ratio, the cost tends to be fixed. Therefore, the product owner is trading off prioritized features to meet the established increment and release dates.

INCREMENT LEVEL DECISIONS

Agile teams are designed to be cohesive and are empowered to make day-to-day decisions. A large part of empowerment in Agile means that the team makes the decisions, not the project manager. Establishing that culture involves knowing which decisions are best made locally and which should be made at the enterprise level. Some ways to guard against the risks of team-based decision making include:

- Implement anonymous or other voting methods that encourage equal weighting of opinions so the most vocal team members won’t always hold sway.
- Change the team or challenge the thinking (e.g., more outside access to work areas) to avoid group think, a byproduct of the synergy and loyalty to one another that are a team’s greatest qualities. If a team gets too insular, advice from outsiders is rejected; facts that support the plan are acknowledged and facts that do not support the plan are ignored.
- Form separate groups under different leaders to propose solutions to the same problem.
- Appoint a role on the team to play devil’s advocate.

7. THE NEED TO FUNDAMENTALLY RETHINK IMPLEMENTATION STRATEGIES

By their very nature Agile programs are likely to produce end products at significantly different points in time. There is often value in moving pieces to implementation rapidly rather than waiting for “the whole” to be ready to launch by conventional means. This can produce a need for very different roll-out strategies, communications tactics, workforce readiness initiatives, etc. It also produces an environment where some implementation activities begin long before other related development initiatives have matured. How does this complicate implementation planning and management?

The case studies dealing with oil and gas, banking, and software development in this report cite the urgent business need to improve the predictability of getting products to market. Here we use time-to-market as the generic term for releasing functionality to customers and/or operations. The mantra in those organizations is integrate early and often!

A number of tailored practices are emerging to accommodate incremental implementation:

- Continuous integration supported by automated builds (for software) and testing.
- Treat compliance, security, and privacy needs as requirements during development.
- Release planning.
- Small and frequent releases/batches.
- Prioritize user stories.
• Coordinate release schedule with all stakeholders (in addition to the product owner).
• Automate wherever possible.
• Be proactive in getting customer feedback on the features and the implementation plan; use techniques such as the scrum retrospective with customers.
• Maintain release management dashboard.
• Hold quarterly release meetings to choose coherent groups of features that span business functionality, customer segments, or geography.
• Flag features with security implications.

Testing and integration create a number of management, quality, and security concerns for Agile projects in many domains, including technology.

Policy and procedures to support Agile methods have yet to be promulgated. Within the Department of Defense for example, acquisition rules—Capabilities Development Process, DOD policy 5000.2, FAR Compliance, Security C&A, Interoperability Certification, etc.—constrain Agile practices. These rules and policies tend to save the testing for last, especially the operational tests. And this practice is not specific to DOD or government agencies. Testing has become accepted practice at the end of the process. Emerging practices include the testers working in the core teams to do functional test planning early.

The test and evaluation community needs to be included and educated about Agile integrated testing, evaluation, certification, and integration. The capabilities development, program management, operational testing, security, and interoperability communities have long been segregated. To gain the benefits of Agile, integrated test teams and continuous integration frameworks must be implemented. The BA or combined BA/tester can plan for smart testing by understanding user stories, functional and nonfunctional requirements, critical paths, and dependencies to identify and manage risk.

Many organizations lack a robust development and test and integration environment. Ideally, this environment would represent operational and security architectures, legacy Interfaces and systems, and the ability to test in a relevant cyber threat environment. Organizations may decide to use early iterations (also known as “sprint zero”) to build up the project infrastructure and procedures in a test environment.

Large batch development means the organization is implementing once, or at least infrequently. Large batch implementations tend to carry big overhead costs, primarily because the siloed teams are waiting for the developed product to be passed to them for action. Developers are waiting for a design package, and the testers are waiting for completed development. The success of any small batch process, including Agile, is the ability to keep the transaction costs low and teams fully utilized.

8. THE NEED TO STILL ENSURE ADEQUATE “SUSTAINABILITY” OF THE RESULTING PROCESSES AND SYSTEMS

Agile programs, by their nature, often skip past some of the more rigorous documentation steps that are fundamental to a typical SDLC methodology. But without reasonable attention to documentation, change management, and ongoing governance methods, those systems and processes can rapidly become increasingly difficult to sustain and evolve to meet changing conditions and requirements. PMO structures for Agile programs need to ensure these
fundamental needs are addressed in a way that supports but doesn’t overly burden the progress of development teams. What are the best practices for dealing with those requirements?

PORTFOLIO LEVEL

An Agile engineering approach provides the portfolio manager the perspectives needed to “fail early and fail fast.” Portfolios carry a degree of risk, and the primary function of the portfolio manager is to manage that target risk given the enterprise goals. Portfolio managers need to prune projects that are underperforming or no longer fit with enterprise strategy. The information that flows from Agile projects provide the portfolio manager with insight on progress and performance much earlier, based on increments rather than final product delivery. The PMO and enterprise PMO with portfolio management responsibilities therefore has a vested interest in the success of Agile PM.

In large organizations, not all projects will be developed using a defined Agile method. For a variety of valid reasons (size, complexity, safety, and criticality), the PMO will need to reconcile and support various development models. The portfolio, as part of the PMO or EPMO, has to be able to assimilate multiple kinds of project progress tracking. See table 3.

One role of the PMO is to maintain enough governance and compliance (depending on industry) so that enterprise risk is managed, while shielding the development and engineering teams from compliance work.21

DOCUMENTATION

Common benefits of Agile are the minimization of documentation and efficiency of self-organizing teams. Against these ideals is the need of the PMO to ensure the products are sustainable well into the future and that other parts of the organization are not duplicating functionality. Well-written documentation supports organizational memory effectively, but is a poor way to communicate during an increment or even a project.22 Some documentation is required to test, train, and support the product throughout its life cycle. And the PMO has governance needs that require automated tracking. The most common Agile method is scrum (see the methodology section of this report), which emphasizes graphical and visual planning and progress tracking by the core team. Simultaneously the scrum master is maintaining an online version of the documentation—one of many Agile PM methods. The PMO should strive to shield the development teams from unnecessary documentation demands, limiting formal documentation to what is needed to sustain the product during operations rather than intra-project team coordination.

The PMO has the responsibility to standardize and educate the increment teams on the tools and techniques used across the enterprise. Standardization may seem foreign to the Agile mindset, but is critically important to facilitate self-organizing teams. Without standards, teams would spend as much time defining working methods as developing projects each time a new team was established.

The tools section of this paper reviews the most common applications to manage projects in an Agile environment. The PMO helps select and implement the consistent use of those tools across the enterprise to support release planning and forecasting, backlog management, increment planning, and monitoring.

<table>
<thead>
<tr>
<th>USE AGILE WHEN...</th>
<th>DO NOT USE AGILE WHEN...</th>
</tr>
</thead>
<tbody>
<tr>
<td>Qualified, well-seasoned, well-disciplined team members will be staffed.</td>
<td>Work is commoditized and can be delivered by lower-skilled staff.</td>
</tr>
<tr>
<td>Interfaces/interfittings are well-defined.</td>
<td>Inexperienced individuals are principal candidates.</td>
</tr>
<tr>
<td>Deliverables can be reasonably distributed in work packets achievable in 2–4 week periods.</td>
<td>Deliverables cannot be reasonably distributed in work packets achievable in 6-week periods.</td>
</tr>
<tr>
<td>Responsiveness to customer requests defines success.</td>
<td>Success is defined as absolute adherence to fully scoped customer requirements identified completely in advance of development.</td>
</tr>
<tr>
<td>The customer or customer representative is available for close collaboration throughout the project.</td>
<td>The customer is not available for close collaboration throughout the project.</td>
</tr>
<tr>
<td>Scope can be adjusted to fit schedule.</td>
<td>Critical (or many) steps involve long lead times or lots of specialized resources.</td>
</tr>
<tr>
<td>The customer is able to reprioritize and add requirements as they progress.</td>
<td>Incremental results have little or no significant value for anyone.</td>
</tr>
<tr>
<td>Work is ground-breaking with steps defined by progress resulting in estimates that are not expected to be reliable.</td>
<td>Process is best implemented as linear (waterfall model in software) or spiral.</td>
</tr>
<tr>
<td>Incremental results have significant value.</td>
<td>Contract requirements or regulations mandate the use of a specific life-cycle process and attendant documentation that does not follow Agile tenets, and under which no ready allowance can be made for an Agile approach.</td>
</tr>
<tr>
<td>Process is by nature iterative, allowing for cumulative results through sprints, and for overlapping planning for the next sprint while work takes place in the current sprint.</td>
<td></td>
</tr>
<tr>
<td>No contract requirements or regulations preventing the use of Agile or making its employment impractical apply.</td>
<td></td>
</tr>
</tbody>
</table>

Table 3. Using Agile22
CUSTOMER

The PMO has to consider what must be addressed in sustainment. Candidate considerations include a help/support desk, continuity of the project team to include the development team, customer training, service levels, enhancements and upgrades, and problem reporting and resolution.

CONCLUSION — DO MORE WITH LESS

One of most frequently cited benefits of an Agile approach is the rapid delivery of functionality to customers. And one of the most critical success factors for Agile is the insistence on collaboration and customer prioritization of features. The Standish Group Chaos™ report has shown that across all projects as much as 64 percent of developed software code is rarely or never used (figure 4). One theory is that the traditional methods force the teams to develop the requirements documents early, before they are really sure what is needed or how it will be used. Customers include everything during this phase under the reasonable assumption that if they leave a requirement out, it will be impossible to get it back into the design.

With Agile, a broad product backlog is established with only the highest priority requirements developed; the backlog is burned down over time. A powerful anecdote is the Agile team that had completed 40 percent of a project’s high-priority requirements when the customer said, “Let’s stop—this is good enough. I’ve got 80 percent of what I need and the rest isn’t worth the cost.” While the Chaos data is based on surveys of IT projects, one can easily imagine the risks and extra time and costs associated with trying to define all customer capabilities, in detail, at the outset of a large program. Remember the requirements uncertainty principal: requirements will not be completely known until after the users have used them!

Iterative development methodologies have demonstrated that small batches offer improved performance over large batches for two main reasons. The first reason is that you get feedback with each batch so you get to adapt and adjust to changing situations. When you only have one large batch, as in a traditional waterfall software development process, you only have one chance to get it right. Developers and clients are rarely that accurate, and even if they are, the chance that the problem they were trying to solve is still exactly as described is very unlikely. The second reason that small batches have improved performance is that the incremental delivery of software gives the users the chance to incrementally create value from it.

Like all efforts to improve performance and efficiency, the biggest gains are those that bridge organizational silos. The intent is to leverage the strength of those silos while supporting communication and development of products or projects that span the organization. It is no wonder many that Agile efforts take on a Lean view: to reduce waste, the PM environment optimizes at the organizational level rather than the sub-business unit. With Agile, the focus of the project manager and program management practices shifts from an inward view that limits change and uses cost and schedule as primary yardsticks to understanding the enterprise’s value chain. Agile project teams become self-directing through intense collaboration with stakeholders. The role of project and program managers is to create the clear vision for the organization and the project, provide resources, and anticipate and remove obstacles.

**Rates of Feature Usage in Software Projects**

- High: 7%
- Medium: 13%
- Low: 16%
- Rarely: 19%
- Always: 45%
- Occasionally: 20%
- Rarely or Never: 64%

![Figure 4. Rates of Feature Usage in Software Projects](image-url)
The benefits of Agile project management are derived in part by placing a tremendous degree of responsibility and accountability on the individual team members. There is an understanding that great teams build great software and those teams should be trusted and empowered to deliver. A good Agile project manager enables the team. He or she helps the team stay focused on the larger business issues and removes obstacles that impact its ability to deliver. The focus is on the team because ultimately the team is on the hook for delivery. Because Agile teams are self-organizing and empowered, the Agile PM focuses more on leadership than in a traditional development environment. Skills such as facilitation, coaching, and team building become key components for project success. The PM is responsible for creating a culture of empowerment and trust and an environment where individuals are motivated to contribute to the team’s success. Agile PMs focus less on assigning tasks and managing the plan and more on maintaining the structure and discipline of the Agile team; trusting that through visibility, inspection, and adaptation the team will deliver the desired results.
AGILE SURVEY RESULTS AND ANALYSIS

BACKGROUND AND METHODOLOGY

Robbins-Gioia distributed a survey during the Agile DC conference held in October 2012. Subsequently the survey was available online and during Agile training classes and sent to clients. The survey was open for approximately one month. The intent of the survey was to capture the respondents’ understanding of Agile methodologies and see if they used it in their professions.

RESULTS

The twenty-six survey respondents varied in occupation and industry. Occupations included scrum masters, Agile project managers, scrum coaches, developers, and consultants.

Various industries were represented:

1. IT – 42%
2. Financial – 23%
3. Government – 4%
4. Management Consulting – 4%
5. Media – 7%
6. Other – 19%

The survey consisted of six multiple choice questions, with four options from which to choose as an answer, three prioritization questions, and one open-ended question. One of the prioritization questions allowed the respondent to provide an open-ended response. Questions asked were:

1. What percentage of your projects uses Agile methods?
   Forty-two percent of respondents selected >75% of their projects use Agile methods.

2. We have adjusted our planning, progress, and reporting approach to align with Agile.
   Thirty-eight percent of respondents agreed with this statement.

3. How long has your organization been using Agile?
   Forty-two percent of respondents said their organizations had been using Agile for at least 6 months, but no more than 2 years.

4. Please rate your organization’s knowledge and skill level using Agile methods.
   Fifty-two percent of respondents reported their organization’s knowledge and skill level using Agile as intermediate.

5. Are you using Agile methods in non-IT development projects?
   Thirty-eight percent of respondents said their organizations were using Agile for non-IT projects.

6. Was there clear executive sponsorship in the transition from the previous methodology to Agile?
   Sixty-nine percent agreed that they had received clear executive sponsorship for transitioning to Agile.

7. Please prioritize the biggest challenges in transitioning to Agile.
   Respondents said that collaboration and including self-directed teams were the biggest challenges in transitioning to Agile. Other answers were: cost effectiveness, organizational politics and silos, and “getting customers steeped in the waterfall culture to be responsive participants in an Agile culture.”

8. Please prioritize key benefits of using Agile on IT projects.
   Respondents ranked faster releases/time-to-market as the greatest benefit of using Agile methods on IT projects. Other answers included improved team communication, a stronger relationship with the end user, and cost effectiveness.

9. Please prioritize key benefits of using Agile on non-IT projects in your organization.
   Faster results ranked high as the key benefit of using Agile on non-IT projects. Next, in order, were cost effectiveness, stronger relationship with end user, and improved team communication.

10. How does PM need to change to apply Agile?
    Best practices for transitioning to an Agile approach to project management included:

    • Adopt Agile PMO mindset.
    • Enhance engagement skill.
    • Embrace the uncertainty of Agile requirements development; it’s ok not to know everything up front.
    • PM must become 1) facilitator instead of director, 2) willing to accept that plans must change on a regular basis, and 3) a proponent of the fact that lowest priority work will probably not get done.
ANALYSIS

Almost all the respondents indicated their organizations have used an Agile approach for development with just under one-half of all respondents indicating they use an Agile approach on three-fourths or more of all projects—overall, a significant rate of adoption. However, three-fourths of respondents reported two years or fewer of experience with Agile. Since many new processes require 3-5 years to become proficient, it is premature to make final judgements on best practice adoption.

Fifty-two percent rated their organization’s knowledge and skill level in using Agile methods as intermediate. Another thirty-three percent said their organizations have a novice level of understanding and experience. Only thirty-eight percent of the respondents indicated their organizations had adjusted their approaches to planning, progress, and reporting to align with Agile methods.

Another factor to consider is the apparent knowledge/skills gap is the use of Agile methods on non-IT projects. Sixty-two percent of respondents said their organizations did not use Agile on non-IT projects. Perhaps a lack of an ability or an understanding for how to perform non-IT work using an Agile method is at the root here.

Organizational politics and a preference for the waterfall method might also explain the gap even though sixty-nine percent of respondents said they had received clear executive sponsorship for transitioning to using an Agile methodology. These two challenges help explain a couple of the responses to how PM needs to change.

CONCLUSION—PROJECT MANAGEMENT CHANGE FOR AGILE

The results from this survey indicate that most respondent organizations have at least experimented with Agile even though they may not be employing Agile methods fully. Yet real success with transitioning to an Agile approach happens when an organization emphasizes throughput and includes customers during trade-off decisions. This is especially important for organizations choosing to apply Agile to non-IT projects. It is necessary then, that organizations consider training program managers, teams, and stakeholders to learn and experiment with PM practices adapted to align with Agile concepts. As respondents to the survey noted, implementing Agile methods must flow from the executives to project sponsors through the PMO to the development teams. Additionally, the respondents noted PMs must become advocates and change agents of the project/program management process as much as they are advocates of the project.
The top 5 benefits of using Agile

- Faster releases/time-to-market
- Stronger relationship with end user
- Cost effectiveness
- Faster results
- Team communication improvement

The top 4 challenges in transitioning to Agile

- Collaboration, including self-directed teams
- Organizational politics and silos
- Reporting progress to stakeholders
- User story creation and management

How does PM need to change to apply to Agile?

- Retraining must be provided for program managers, teams, and stakeholders to learn and experiment with PM practices modified to align with Agile concepts.
- Change is needed top-to-bottom in the organization, flowing from executives to project sponsors through the PMO to the development teams. The PM must become an advocate (change agent) for the process as much as the project.
- Embrace the uncertainty of Agile requirements development. It’s ok to not know everything up front.
- PMs must become 1) a facilitator instead of director, 2) willing to accept that plans must change on a regular basis, and 3) a proponent of the lowest priority work that will probably not get done.
- Real success happens at a tipping point when the enterprise adapts to a mindset that emphasizes throughput and brings customers into trade-off decisions.
SUCCESSFUL APPLICATIONS OF AGILE: CASE EXAMPLES

The eight management practices discussed in the “Program Management for Agile: Perspectives” chapter originate from our research and experience, including practitioner surveys, tool and methodology reviews, published works, three decades of working in all aspects of program management, and the following case studies. The table below relates these case studies to different practices explored in the “Perspectives” chapter.

<table>
<thead>
<tr>
<th>PROGRAM MANAGEMENT PRACTICE</th>
<th>TECHNOLOGY SERVICES</th>
<th>FINANCIAL SERVICES</th>
<th>OIL AND GAS</th>
<th>U.S. GOV’T AGENCY</th>
<th>U.S. REGULATORY AGENCY</th>
</tr>
</thead>
<tbody>
<tr>
<td>ADJUSTED PROJECT, PROGRAM, AND BUSINESS ANALYSIS MANAGEMENT PRACTICES</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>•</td>
</tr>
<tr>
<td>RAPID INCREMENTAL DECISION-MAKING APPROACH</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>•</td>
</tr>
<tr>
<td>FRESH APPROACH TO IMPLEMENTATION</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>•</td>
</tr>
<tr>
<td>ADEQUATE “SUSTAINABILITY” OF THE RESULTING PROCESSES AND SYSTEMS</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>•</td>
</tr>
</tbody>
</table>
TECHNOLOGY SERVICES COMPANY’S TIGHTER COLLABORATION IMPROVES PLANNING, TIME TO MARKET, AND PRODUCT QUALITY

ORGANIZATION

Widely recognized for its achievements in the financial services, education, and public sectors, a large software and technology services company on the Fortune 500 list serves 25,000 customers in more than 70 countries. The company’s entrepreneurial culture drives it to acquire and integrate complementary businesses, offering complete solutions to ensure client success.

ENVIRONMENT/CHALLENGE

While the overall company professes an entrepreneurial spirit, its software development process has been dormant the last few years. The firm had been functioning under a standard waterfall model, which did not actively engage necessary customer input throughout the product development cycle. Deadlines were missed and morale plummeted, calling the value of the waterfall model into question. In addition, management prescribed that more than 60 percent of software development be completed by offshore resources. Roughly 6 out 10 projects were either delivered late or with considerable quality issues.

WHY AGILE?

Clearly, the company needed to do something to turn software development—its bread and butter—into a profitable enterprise once more. An employee who was also an Agile practitioner was called in to transition leadership from the current waterfall to the Agile process. As is typical with any new initiative, changing people’s mindsets was the biggest impediment to Agile. First, the practitioner educated employees in each of the four pertinent departments over a six-month period on the use and importance of Agile. Over the next three months, he held a series of virtual and on-site training sessions on the specific goals of Agile and role of each person in the process. All in all, the practitioner trained 55 people within the company.

Initially, there was serious skepticism about the new process, but its benefits—particularly speed to market—were finally realized. Scrums were established—small cross-functional teams—to ensure the project moved quickly and adapted to changes in scope. The scrums implemented Agile techniques such as daily standups to identify progress and areas for collaboration and improvement. The Agile process enabled people to work together more closely, resulting in swifter completion of development phases.

LONG-TERM BENEFITS FROM AGILE

The practitioner had initially piloted the Agile process for a smaller project in one of the company’s overseas offices. He reviewed and analyzed the pilot results with the management team that would ultimately decide on the implementation of Agile. Each of the four departments in the company was then asked to choose a pilot project; the practitioner stepped in as scrum master for their projects. Each group realized significant improvement in cycle time and project quality. For example, the group with the strongest results recognized a week savings in time on a sprint, producing software faster and with better results for its clients. Through the use of sprint retrospectives the teams tailored their practices. Subsequent sprints reflected a shallow learning curve with rapid improvement in quality and velocity. Key areas of process focus were:

- The teams learned new approaches in dealing with change from clients and customers including earlier and more frequent touch points using visual aids.
- Initially, the scrum teams were very senior. This was a huge benefit in that the teams learned to change their perception about accepting change and executing off the change.
- The scrums took over communication in real time through daily standups and close working configurations, resulting in a growing acceptance of the need for more timely communication.
- There was a considerable reduction in administrative overhead, such as email as the use of the daily scrum implemented a strong sense of accountability in the team.
- The planning done in sprint planning was actually more robust than in previous project planning sessions not using Agile.

The following benefits were seen in implementing the process:

- Quality issues were reduced by roughly 25 percent.
- Because they were more directly involved, clients and customers overall were very receptive to the new process.
- The majority of sprints were brought in greater than 10 percent faster of the original time scheduled.

CASE EXAMPLES
BANK RETHINKS PMO GOVERNANCE AND RELATED PM PRACTICES AS PART OF THE TRANSITION TO AGILE DEVELOPMENT

ORGANIZATION
A large regional bank serving families and businesses in nearly 200 locations across a Southeastern state is committed to delivering products and services that add value and convenience to the lives of its clients, supported by the latest technology. While proudest of its awards for community relations and maintaining a family-friendly work environment, the bank was also recognized for being an innovative user of technology.

ENVIRONMENT/CHALLENGE
The bank asked a software development company that it contracted with to improve a back-office processing application for 100 key users. The contractor developed the software and attempted to implement the application; however, the process proved challenging. The application required significant customization and therefore necessitated continuous involvement with bank users—and the current software development system was not enabling that interaction to happen. As a result, the bank was disappointed with the application’s implementation and, ultimately, its efficiency.

WHY AGILE?
In an effort to turn things around, the original project manager called on an Agile practitioner and expert at the contracting company’s main office to step in and apply Agile to this project.

First, the practitioner helped define the Agile PMO project management solution by aligning the PMO’s governance to accommodate business requirements and leadership priorities. Scrums were established—small cross-functional teams—to ensure the project moved quickly and adapted to changes in scope. Five teams with 25-30 people on each were responsible for different sections of the product, such as cash, securities, trading, etc. The scrums held to Agile techniques such as daily standups to identify progress and areas for collaboration and improvement.

Agile empowered the small, highly skilled teams to increase productivity through an iterative approach, producing smaller pieces of functionality more frequently, resulting in more flexible, user-focused results. The bank was involved throughout the project life cycle and benefited from the project value at frequent milestones, as opposed to only at project completion. This meant that changes could be made quickly and easily to ensure the final product was exactly what the bank required. Implementation was a fluid process where customer requirements gathering ran the course of software development.

The implementation activity was planned based on the initial business review of the client’s current processing. This planning was then adapted to divide the implementation into various sprints. The first sprint was a design review for the functionality requested by the client; remaining sprints were planned for development in each section of the overall product.

LONG-TERM BENEFITS FROM AGILE
The bank recognized substantial efficiencies in software development by shifting from a waterfall development process to an Agile methodology. The scrum teams were deeply engaged with the client and able to quickly create and deliver software changes in a 24-hour turnaround. Other benefits included:

- Identification of software issues much earlier in the life cycle.
- Culture change in favor of fresh thinking regarding development approaches, which carried over to other projects.
- Reduced backlog because of the efficiencies gained by the Agile approach, which positively influenced the organization’s acceptance of Agile.
- Close client involvement with testing and validation, positioning this 6-month project for success.
ENERGY COMPANY CREATES A LIFE CYCLE VISION FOR NEW PRODUCT DEVELOPMENT WITH AN AGILE APPROACH

ORGANIZATION

Oil and gas companies face greater challenges as costs to operate continue to rise and reserves decrease. Because new product development is essential to remain competitive, one large energy supplier planned a global rollout of a software initiative to speed time to market. Given the unique characteristics associated with software development in the energy supplier’s commercial software organization and its respective business units, it was essential that it engage a transformation advisor that had a deep understanding of software development and technology, process, strategy, governance, and the successful management of a change of this magnitude. The primary objective is to establish a broad product life cycle accommodating multiple development models. For many products, including software intensive products, the preferred development approach will be Agile.

ENVIRONMENT/CHALLENGE

A gated new product development (NPD) process was developed and implemented for all new products for the energy supplier’s system globally. A toll-gated NPD process is a parallel software development effort that complements the client’s broader process, specifically focused on helping to ensure the quality, efficiency, and effectiveness of the client’s efforts worldwide.

The transition team began by laying the foundation for the discovery process, including a structured approach to interviewing management in all business units. As the project moved from discovery into the pilot phase, the team played a lead role in developing the foundational materials, templates, and artifacts associated with the process. The team was instrumental in developing critical assessment tools that allowed the supplier to clearly identify, understand, and plan for the challenges to implementation.

WHY AGILE?

One of the key instruments designed was the Agile assessment and maturity model. This tool enabled the quantitative and qualitative evaluation of the Agile readiness of each of the business units in the client’s organization. The assessment enabled identification of gaps between the business unit’s current state and the level needed to effectively implement the key Agile-driven components of a toll-gated NPD process, including role coverage, expertise, and process maturity. From this gap analysis, strategies were formulated to address shortcomings in capabilities to use Agile, including skills training, coaching, and instruction on the Agile development process.

A global rollout of a process like this involves significant organizational change. The transition team developed the change management strategy for this software development process that included the design of an organizational readiness survey and an executive level organizational change management questionnaire. The transition team assigned “change champions” from each of the business units and conducted change champion training and a risk management for project managers workshop. Throughout the transition, the team provided advice and direction on establishing communities of practice and other developmental and communications approaches, tools, and strategies critical to the effort.

The transition team stayed with the effort by developing the data collection tools and being on point for the analysis of this data and was responsible for developing the implementation plans and working closely with the organization’s management and business units on implementing those plans.

As part of the NPD life cycle, a new governance model was designed for the toll-gated NPD process that helped ensure the sustainability of the plan going forward from being simply a “project” to a way of doing business. Through the use of early organizational readiness “discovery” sessions and sprint retrospectives the teams tailored their project and program practices. Subsequent sprints reflected a shallow learning curve with rapid improvement in quality and velocity. Key areas of process focus were:

- An Agile approach was used to generate the initial NPD objectives, components, and templates for transition team use, and define and design NPD process changes. The key elements of that approach are formation of sub-teams of program managers to support document and process change definition and design; sub-teams focused on specific documents and processes based on expertise; and iterative requirements development.
- Sub-teams developed requirements collaboratively in two or three one-week cycles. Upon completion of the requirements, the teams obtained approval from the broader program manager community.
• Iterative document and process change design was executed using the same approach; again, approval of the broader PM community was obtained upon design completion.

Application of this process led to shorter overall cycle times and improved alignment of outcomes.

Another example of Agile application was in the development of a SharePoint infrastructure to support project and program execution within the NPD framework. Independent business units planned to use the SharePoint infrastructure, which drove the need to perform numerous iterations involving all of them to create a single infrastructure that would be unanimously acceptable. This was accomplished by applying:

• Rapid prototype development, which occurred in one-week cycles. After each cycle, the prototype was made available to key stakeholders in each business unit, and a meeting was held to walk through the prototype, demonstrating functionality and identifying needed features.

• Pilot testing over a four-week period, with similar one-week cycles and quick revisions of the SharePoint functionality during each cycle.

• Rolling out the high-value SharePoint infrastructure to the business units, who were familiar with the functionality based on the development approach, leading to a high adoption rate and a flatter learning curve.

Because of the need to coordinate with multiple business divisions, client representatives, PMO program and project managers, and IT scrum masters, governance was a major component of the new approach. For example:

• Daily IT scrum meetings were incorporated into standard PMO communication plan templates.

• PMO program managers became involved in daily IT scrum meetings to capture status, issues, and risks affecting related program components (i.e., different scrum teams).

• Agile concepts were incorporated into many of the PMO processes and tools to take advantage of iterative delivery approaches, particularly around the integration and testing of product components. More unit testing was pushed to the increment teams.

LONG-TERM BENEFITS FROM AGILE

• Improved quality, efficiency, and effectiveness of software development internationally.

• Development of assessment tools that allowed clear identification and understanding of client challenges and the ability to plan for those challenges.

• Greater process sustainability.
U.S. AGENCY LOOKS TO TEAMWORK AND COLLABORATION TO IMPROVE CUSTOMER SERVICE ORGANIZATION

Last year, one of the U.S. government's largest and most complex organizations welcomed more than 300 million people into the United States. Besides receiving people into the United States, the agency ensures that dangerous people and materials do not enter, along with facilitating lawful international travel and trade.

To help accomplish its mission, the agency initiated a multi-year project to modernize the business processes essential to securing U.S. borders, speeding the flow of legitimate shipments, and targeting illicit goods. The project will result in a commercial trade processing system that consolidates and automates border processing, ultimately becoming the single window for all trade and government agencies involved in importing and exporting, improving efficiency and security. It will facilitate collecting and distributing standard electronic import and export data.

The project has been underway for more than 10 years and deployed key functionality such as the secure data portal, periodic monthly statement, electronic manifest, and entry summary processing (including post-summary corrections). Companies that use the system can track their import compliance history and manipulate data to analyze their trade volumes and performance, as well as pay duties on a periodic basis. Ultimately, it will serve as the common portal for importers and exporters to file trade data with federal agencies and then access and analyze that data on an account-level basis, instead of filing separate documentation for each agency.

ENVIRONMENT/CHALLENGE

In 2012, the program went into its planned operations and maintenance phase, but much functionality was still needed. The agency conducted a series of investment reviews to assess the program. The result was the identification of number of action items involving close coordination of the agency with its stakeholders—40 other government agencies, importers, exporters, and brokers—to develop and implement the remaining functionality. The agency decided to use customized Agile methodology to help deliver functionality to its end-users quicker and meet changing business needs.

The roots of the agency's adoption of Agile start with the Federal CIO's 25 Point Implementation Plan to Reform Federal Information Technology Management, which includes a mandate to use modular approaches to incrementally develop and deliver IT systems. In implementing this mandate, an integrated product team (IPT) was chartered in August 2011 to develop a framework for Agile software development. The Agile IPT subsequently drafted a white paper detailing the framework along with guidance to use it as the starting point for implementing Agile. Accordingly, the agency used the white paper as the basis for its Agile implementation and to gain the benefits of faster time-to-mission-value, reduced project risk, and reduced costs.

Agile empowers small highly skilled teams to increase productivity through an iterative approach, producing smaller pieces of functionality more frequently, resulting in a more flexible, user-focused development process. The customer is involved throughout the project life cycle and value is shown at frequent milestones, as opposed to only at project completion. This means that changes can be made quickly and easily to ensure the final product is what the customer wants and needs.

WHY AGILE?

As mentioned above, a number of action items were identified to necessary to deliver the remainder of the system's capabilities on a continuous, frequent basis. Such a transformation would leverage Agile software development and program management methods to dramatically increase the agency's organizational responsiveness and performance and search for efficiencies in a very constrained fiscal environment. To quickly address the action items, the agency established the scrums for the Agile transition plan, Agile PMO structure, governance model, product backlog, future release plan, out-year funding strategy, and legacy system migration. Robbins-Gioia staff members co-led or actively participated in and collaborated with all scrums. Specifically, Robbins-Gioia helped define the new Agile PMO's governance to accommodate business requirements and leadership priorities. The RG team tailored the software engineering life cycle embedded in policy to fit the agency’s Agile process and serve as the engineering and investment control basis for the Agile plan, and defined a PMO structure that could effectively implement it. Robbins-Gioia acted as scrum master on multiple teams facilitating sprint ceremonies including release planning, sprint planning, daily stand-ups, sprint reviews, and retrospectives.
Scrums were established—small cross-functional teams—to ensure the project moved quickly and adapted to changes in scope. The scrums held to Agile techniques such as daily standups to identify progress and areas for collaboration and improvement. One of the Agile PMO’s main missions was enabling the teams to execute by removing administrative burdens from them.

The other mission was to maintain the product backlog—the body of user requirements—as expressed in user stories. The user stories and consequently the backlog change almost continually because of new laws, legislative mandates, mission needs, and requirements from diverse user and business communities—which is comprised of other government agencies as well as importers, exporters, and brokers. The PMO must constantly revitalize the backlog to reflect the business communities’ needs at the moment and make sure they are satisfied. This very demanding and difficult job is suited to the responsiveness and fluidity of the Agile methodology. Agile planning techniques demand focus on high-priority requirements. Requirements are not uniform from top-to-bottom. The highest priority requirements get the highest level of attention in decomposing them. Low-priority requirements may remain as simple user stories in the product backlog.

As the Agile development team pushes high priority requirements through its software factory to create capabilities and convert the product backlog into reality, the business stakeholders have the opportunity to continuously refine and reprioritize the product backlog. A capability product owner is embedded with each development team to interface with the business and product management committee so that the business always drives what the development teams are building and validates their results.

Before rolling out the methodology enterprise-wide, the Agile PMO will begin Agile execution in early 2013 with a four-month pilot project consisting of three Agile teams working in parallel to collectively produce working software. This effort will require putting in place the organizational capacity to perform, which means training and applying lessons learned.

LONG-TERM BENEFITS FROM AGILE

- A laser-like focus on building value for the business as determined by the business.
- Improved trust between technical and business communities as a result of continuous delivery of value.
- Ability to respond to ever-changing business needs.
- Increased customer satisfaction.
U.S. REGULATORY AGENCY CHANGES IMPLEMENTATION AND SUSTAINMENT METHODS IN THE WAKE OF AN AGILE DEVELOPMENT APPROACH

ORGANIZATION

A large federal regulatory agency needed to deliver its products more quickly and meet the requirements of its internal customers to promote safety and security.

ENVIRONMENT/CHALLENGE

The agency’s current maintenance operation and modernization development culture lends itself to Agile, with its customer-focused ethos to make needed modifications quickly. But integrating Agile-based development with longer-term support is challenging because of the differences in definitions. For example, “development” as described by some customers was really sustainment. Other areas used the term development more properly to describe new capabilities.

As part of the transition to Agile, clear swim lanes have been established to separate solution development from sustainment. Integration between the two swim lanes is specified; they are organized in a staged gate format with clear hand-offs to sustainment, support, operations, and maintenance. In-house staff members are taking the roles of project managers and business analysts to create a cohesive team structure.

WHY AGILE?

The agency is formally matching project managers with business people in the customer offices to ensure a smooth Agile transition, better requirements definitions, and better results. A new business analyst position in the agency’s IT office is in being ramped up to make sure this process works well.

Through the Agile process, communications have been strengthened with customers right up front because of their necessary involvement in the process. While Agile may take more of the customers’ time in the beginning, it is proven to reduce rework as the projects become operational. The agency is issuing a directive, complete documentation, and a project manager’s handbook on the new process and to make sure that there is organizational understanding.

Likewise, the old program manager training is insufficient. The agency is building on their previous experience with rapid application development and joint application development from the 1990s as a basis for this training.

To involve employees in the new Agile initiative from the beginning, the agency stressed the “WIFM,” (what’s in it for me) principle by demonstrating that higher-quality, faster results would result from better requirements development in the Agile process. Then the agency created integrated teams and a communications channel to management, so that everyone in the executive ranks understood the changes and the value.

The agency now has standard contracts that are differentiated between requirements and development and maintenance. In statements of work, the agency specifies Agile tools and techniques that must be used. Separate vendors are sometimes used to enforce the hand-off between stages.

LONG-TERM BENEFITS FROM AGILE

The following benefits were seen in implementing the process:

- A reduction in quality issues by roughly 25 percent.
- Overall good reception by customers because of their direct involvement with the process.
- Majority of sprints were brought in 10 percent faster than the original time scheduled.
- The teams learned new approaches to dealing with change from the clients.
- Reduction in administrative emails as real-time communication took place in scrums.
AGILE METHOD AND MODEL REVIEW

Since the publishing of the Agile Manifesto in 2001, a number of management and engineering methodologies have been developed and/or refined—some competing with and some supporting Agile methods. Major models have continued to evolve in scope; generally, all have focused on close team interaction and contributing value to the organization. Examples include the Capability Maturity Model Integration® (CMMI), developed and maintained by the Software Engineering Institute, and the Unified Process and its better-known refinement, the Rational Unified Process (RUP), developed by Rational Software (now IBM).

The Agile philosophy shares common ground with these practices in that it was developed to correct and improve conventional management and engineering weaknesses and take advantage of new technology and knowledge. In the broadest sense, Agile is a philosophy and mindset defined by a set of core values and much wider set of principles. Almost all the specific methods used to implement Agile follow these values and principles. Each method goes on to define a particular set of practices. We believe that over the next several years the distinction between the various methods will blur as companies and organizations tailor their own development and project management methodology by picking and customizing from the universe of practices.

Following is a discussion of some of the more common methods in implementing Agile. In the public sector, many organizations have established guidelines and regulations to improve the efficiency and effectiveness of development activity. With the emphasis on understanding functional and customer needs and modular development, they share in some of the Agile value philosophy.

Agile arose from software development but is beginning to be applied to projects outside the world of software. In those cases, the teaching of Deming and others and the formalization of Lean manufacturing applied to knowledge work have dominated in place of methods like scrum and XP.

AGILE

Agile development encompasses concepts previously used in software development. These concepts were documented as Agile themes and principles by 17 practitioners who called themselves the Agile Alliance. In February 2001 the Alliance released “The Agile Manifesto,” (Figure 1) in which the members declared:

“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.”

The Alliance added that while it recognized the value in the second part of each statement (i.e., “processes and tools”), they saw more value in the first part (“individuals and interactions”). The Alliance further delineated their vision with 12 principles as shown in figure 1.

MODULAR DEVELOPMENT FOR U.S. FEDERAL AGENCIES: OFFICE OF MANAGEMENT AND BUDGET GUIDANCE

One approach to reducing the risks from broadly scoped, multiyear projects is the use of shorter software delivery times, a technique advocated by OMB in recent guidance. Specifically, OMB’s June 2010 memo on IT financial system reforms and the December 2010 IT management reform plan encourage modular development with usable functionality delivered in 90 to 120 days. In addition, the Federal Chief Information Officers (CIO) Council, chaired by OMB’s deputy director for management, encouraged the sharing and adoption of efficient IT development practices, such as those in OMB’s IT guidance.

As outlined in the 25 Point Implementation Plan to Reform Federal Information Technology Management, federal agencies have traditionally taken a multi-year “grand design” approach for developing, modernizing, and enhancing IT investments. This approach is based on the notion that requirements must be fully detailed before development work can start. Although this seems like the responsible approach, experience has shown that large, complex IT implementations often encounter cost and schedule overruns, as collecting extensive requirements can take years to complete. Agencies then lose visibility into the performance of these multi-year IT development investments, which affects their ability to implement corrective actions that reduce risk or mitigate financial exposure. Investment risk is further increased because the IT solutions that once addressed agency requirements may no longer be relevant or a priority.

There is a risk that substantial funds are allocated to narrow projects or requirements that are likely to change over following a plan.”


The AGILE Manifesto

The Twelve Principles of Agile Software

1. Our highest priority is to satisfy the customer through early and continuous delivery of valuable software.

2. Welcome changing requirements, even late in development. Agile processes harness change for the customer’s competitive advantage.

3. Deliver working software frequently, from a couple of weeks to a couple of months, with a preference to the shorter timescale.

4. Business people and developers must work together daily throughout the project.

5. Build projects around motivated individuals. Give them the environment and support they need, and trust them to get the job done.

6. Agile processes promote sustainable development. The sponsors, developers, and users should be able to maintain a constant pace indeﬁnitely.

7. Working software is the primary measure of progress.

8. The most efﬁcient and effective method of conveying information to and within a development team is face-to-face conversation.

9. Continuous attention to technical excellence and good design enhances agility.

10. Simplicity—the art of maximizing the amount of work not done—is essential.

11. The best architectures, requirements, and designs emerge from self-organizing teams.

12. At regular intervals, the team reﬂects on how to become more effective, then tunes and adjusts its behavior accordingly.

www.agilealliance.org

Figure 1.
The Agile Manifesto. 9


13 Ibid.

ward outdated solutions without any returns on the investments and agencies can encounter budgetary constraints before substantive work is completed.

In contrast, agencies using modular approaches used for IT development can deﬁne, develop, and deploy signiﬁcant functionality within months instead of several years. OMB guidance highlights critical success factors for designing, resourcing, and managing investments in a modular fashion, including the need to strengthen collaboration among agency IT, program, acquisition, and ﬁnance oﬃces. To ensure successful collaboration across these oﬃces, it is important to have a common and consistent vocabulary for key modular IT and acquisition terms and concepts, describe how the work of each community ﬁts together, and stress the necessity of having integrated project teams. This guidance also provides tools and strategies for agencies to use in adopting modular IT development practices. Finally, this guidance provides excerpts from a sample performance work statement to support modular software development. Agencies are expected to assess and adjust capital planning and investment control and acquisition planning policies and practices to more explicitly incorporate these modular approaches in IT development and acquisition. Using these approaches, agencies can realize faster returns on IT investments and deploy technology solutions that will propel them toward a 21st century digital government that is eﬃcient, eﬀective, and focused on improving the delivery of services to the American people. 10

AGILE CERTIFICATION AT THE PROJECT MANAGEMENT INSTITUTE

Agile is a topic of growing importance in project management. The marketplace reﬂects this importance as project management practitioners increasingly embrace Agile as a technique for managing successful projects. Gartner predicts that in the near future, Agile development methods will be used on 80 percent of all software development projects. 11 PMI’s research has shown that the use of Agile has tripled from December 2008 to May 2011. Further research demonstrates that the use of Agile decreases product defects, improves team productivity, and increases delivery of business value. 12
PMI’S AGILE CERTIFIED PRACTITIONER (PMI-ACP) CERTIFICATION

The PMI-ACP certification recognizes and validates an individual’s knowledge and expertise in Agile principles, practices, and tools and techniques across Agile methodologies. The PMI-ACP certification requires a combination of Agile training, experience working on Agile projects, and examination on Agile principles, practices, tools, and techniques. This global certification also supports project practitioners in meeting the needs of organizations that rely on a diversity of methods to support project management. The key difference between PMI-ACP and other Agile certifications is that PMI-ACP covers knowledge, skills, tools, and techniques across Agile methodologies. Many other certifications only focus on a single Agile methodology.¹⁴

Currently, PMI has no plans to create a PMBOK for Agile because PMI views PMBOK and Agile as complementary frameworks. PMBOK tends to describe what should be done, whereas Agile methods tend to focus more on the “how.”

INTERNATIONAL INSTITUTE OF BUSINESS ANALYSIS AND AGILE

The Agile extension to the Business Analysis Body of Knowledge® (BABOK®) Guide addresses Agile development practices that have emerged since BABOK’s original publication in March 2009. The Agile extension was developed as a collaborative effort between the Agile Alliance and IIBA® (International Institute of Business Analysis) to provide guidance to the effective practice of business analysis in an Agile context.

The IIBA is currently reviewing all submitted comments for consideration in the next revision.¹⁵

SOFTWARE ENGINEERING INSTITUTE (SEI) CMMI AND AGILE

Widely accepted industry guidance on software development has recently been revised to include more Agile approaches. Specifically, the Software Engineering Institute’s Capability Maturity Model Integration updated some process areas to help those using Agile to interpret its practices.

CMMI V1.3 adds Agile, which includes an interpretation guideline and notes for the applicable process areas. Earlier, the SEI published the technical report CMMI or Agile: Why Not Embrace Both,¹⁶ to make clear that CMMI and Agile can co-exist.

BEST PRACTICES FROM THE FEDERAL CIO COUNCIL AND AMERICAN COUNCIL FOR TECHNOLOGY/INDUSTRY ADVISORY COUNCIL (ACT-IAC)

These organizations aggregate best practices from across the Executive Office of the President and promote performance goals such as reduced capability duplications.

The ACT-IAC Emerging Technologies Shared Interest Group (SIG) has the lead on Agile; as of now, there is little published material on Agile practices.

BUSINESS CAPABILITY LIFE CYCLE (BCL) AT DEFENSE BUSINESS SYSTEMS

BCL is the overarching framework for the planning, design, acquisition, deployment, operations, maintenance, and modernization of Defense Business Systems (DBS). The BCL process, tailored to the unique requirements of business systems, facilitates rapid DBS acquisition and deployment. Official BCL guidance, based on industry best practices, studies, and emerging legislation, is published in chapter 12, “Defense Business System Definition and Acquisition Capability Lifecycle Guidebook.”¹⁷

The BCL addresses the challenges facing the weapon systems acquisition process, which is not fast enough to keep up with the continuous introduction of new IT capabilities. Industry and government have learned the “big-bang” approach to delivering IT seldom meets user expectations; the time lag between requirements and delivery is often too long.

BCL improves capability definition and streamlines the acquisition process by:

- Merging duplicate processes and streamlining the governance structure
- Emphasizing rigorous analysis of a capability gap or need and summarizing it in a comprehensive problem statement
- Using a business case and program charter to streamline documentation required by the Office of the Secretary of Defense
- Emphasizing delivery of business capability in compressed time frames of 18 months or fewer.

BCL is modeled to some degree on the Agile concept of iterative capability releases. It requires focused upfront analysis of a problem or capability gap to identify its root cause and determine whether a material (i.e.,


Focus on implementation, not documentation. The challenge many organizations face is the need to implement Agile practices while remaining compliant with overarching laws and regulations. For the DOD there are a number of public laws and department policies that govern how projects are approved, developed, and implemented. The department has developed an implementation model in BCL to extract benefits of an Agile approach without changes to law or policy.

**Project Management Accountability System (PMAS) at the U.S. Department of Veterans Affairs, Office of the Chief Information Officer**

The Program Management Accountability System (PMAS) is a U.S. Department of Veterans Affairs (VA), Office of the Chief Information Officer (OCIO) management approach requiring all VA IT projects to deliver customer-facing functionality on an incremental basis, providing real, usable, mission-delivery capabilities for the staff who directly serve our nation’s veterans. 18

PMAS was designed to reduce risks; institute monitoring, controlling and reporting disciplines; and establish accountability. The system requires all IT programs under its purview to use incremental product build methods focusing on near-term, assured delivery of new capabilities to its customers.

---


that resulted in long delivery cycles. As a result, users had to wait many months and often years to receive new functionality and tools with no guarantee that the delivered product would satisfy their needs. By contrast, Agile is an iterative development methodology not in common use in the federal arena but embraced at many successful technology companies. Agile emphasizes rapid delivery and user involvement from the beginning to ensure successful delivery of tools that the users want and need.

USPTO uses the Agile methodology known as scrum. USPTO selected Scrum in part because of its iterative, incremental process that optimizes project predictability and controls risk by dividing large, complex projects into smaller, time-limited, development increments that are easier to manage.

USPTO will use Scrum to build the Patents End-to-End (PE2E) system. The PE2E technical design will be based on service-oriented architecture principles, which increase the likelihood of developing systems that are adaptable to changing business and regulatory requirements.

An Office of the Inspector General report found that "PE2E lack[ed] prioritized, high-level requirements in its product backlog." Specifically, the report stated:

Although USPTO had a product backlog for the first release of PE2E, the backlog did not include prioritized high-level requirements for the entire
project before development of the first release began. Agile development is an evolutionary process that allows for a wide degree of flexibility; plans and requirements are expected to change. There is a common misconception, however, that long-term planning is not necessary when using Agile methods. In fact, the first steps in Scrum—which is a fitting methodology for large, complex projects such as PE2E—are to define high-level requirements for the entire project and prioritize those requirements based on business and technical value in a product backlog. The requirements are then organized into a schedule of releases. Detailed planning occurs during development of the release. When completed, each release is deployed to end users who then provide feedback used to update the requirements in the backlog for future releases.  

<table>
<thead>
<tr>
<th>Table 1. Required PMAS artifacts for the first increment</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>PROJECT</strong></td>
</tr>
<tr>
<td>PROJECT CHARTER</td>
</tr>
<tr>
<td>IPT CHARTER</td>
</tr>
<tr>
<td>PROJECT QUAD CHART</td>
</tr>
<tr>
<td>ORGANIZATIONAL BREAKDOWN STRUCTURE</td>
</tr>
<tr>
<td>PROJECT MANAGEMENT PLAN</td>
</tr>
<tr>
<td>PROJECT SCHEDULE</td>
</tr>
<tr>
<td>RISK LOG</td>
</tr>
<tr>
<td>STAFFING ASSIGNMENTS SPREADSHEET</td>
</tr>
<tr>
<td>PROJECT DEPENDENCY MATRIX</td>
</tr>
<tr>
<td>PMAS CONTRACT SHEET</td>
</tr>
<tr>
<td>ACQUISITION PLAN STRATEGY</td>
</tr>
<tr>
<td>REQUIREMENTS SPECIFICATION DOCUMENT</td>
</tr>
<tr>
<td>CONCEPTUAL DESIGN</td>
</tr>
<tr>
<td>CONOPS</td>
</tr>
<tr>
<td>PRODUCT EVALUATION AND DECISION ANALYSIS PLAN</td>
</tr>
<tr>
<td>ACCEPTANCE CRITERIA PLAN</td>
</tr>
<tr>
<td>PMAS READINESS CHECKLIST</td>
</tr>
<tr>
<td>PMAS READINESS EXECUTIVE DECISION MEMORANDUM</td>
</tr>
<tr>
<td>PMAS READINESS NOTIFICATION</td>
</tr>
</tbody>
</table>

THE MOST COMMON AGILE METHODOLOGIES

This section summarizes a number of popular Agile methodologies in use today throughout government and industry. Of the Agile methodologies listed, the most widely used are XP, Scrum, and Lean software development. According to Version One’s State of Agile Development Survey 2009, 83 percent of all Agile teams use the following methodologies: Scrum (54 percent), Scrum/XP hybrid (11 percent), XP (2 percent) and Lean software development (2 percent) (see figure 4). These methodologies were selected to fit slightly different objectives. XP is a software engineering Agile process, Scrum is an Agile project management framework that can be used alone or in coordination with any Agile process or processes, and Lean software development is focused on reducing waste, increasing quality, and establishing a culture of continuous improvement. Adopting Agile is not an all-or-nothing, purist-only approach. It is important to match the best approach, mix, or hybrid based on...
the needs of the individual project. See tables 2 and 3 for comparisons among different aspects of different Agile methodologies.

EXTREME PROGRAMMING (XP) is founded on the values of communication, simplicity, feedback, and courage. It emphasizes constant user involvement and technical best practices, including test-driven development, pair programming, and constant refactoring. It works by bringing the whole team together in the presence of simple practices, with enough feedback to enable the team to see where they are and to tune the practices to their unique situation. It is often used together with Scrum to specify lower-level development and team processes.25

Of the three Agile methodologies described in this section, XP is most focused on daily engineering processes for the software team. These include concepts that overlap with other Agile practices, such as Scrum, by employing user stories to define requirements. XP also includes several “extreme” concepts, such as shared code ownership, writing the tests before the code, and developing code in pairs. One important aspect of XP is the use of automated tests that are run each time new code is developed. If new code causes a test to fail, no new code is written until the test passes. This continuous testing approach, to include integration testing, allows errors to be corrected as soon as possible.

XP consists of the following 13 core elements:

1. Whole team: The team is made up of many functional roles that all work together to develop the product. It is not us and them, but we.
2. Customer tests: The customer witnesses an automated acceptance test of the desired feature.
3. Small releases: Development is broken into multiple small releases to decrease risk and provide working software early and often.
4. Planning game: The team continually monitors what is currently being developed to determine what needs to be developed next.
5. Collective ownership: The code is owned by the team, not by individual team members, so everyone can modify the code.
6. Coding standard: A common standard is used for code development so that it is easier to understand and modify.
7. Continuous integration: Integration is not pushed off until the end, but is part of the daily workflow.
8. Metaphor: The team uses a simple way to describe the program vision so that it is easy to understand and communicate.
9. Sustainable pace: The team works 40-hour workweeks, and surges for a short time, if needed.
10. Simple design: The team builds software to a simple design to minimize waste associated with redundancy and rework.
11. Pair programming: Junior developers are paired with more experienced developers for mentoring as well as providing peer review of the code.
12. Refactoring: The code is continually refined and simplified.
13. Test-driven development: XP uses a test-first approach that requires the test to be written prior to code development, and then the test is run frequently as the code is being written. New code is not written until the test passes.

SCRUM is the most commonly used Agile methodology, and often used with the other methodologies listed below. It is based on self-organizing teams, a product owner responsible for managing requirements to deliver the maximum value to the organization, requirements managed through “backlogs,” and a set of process ceremonies (tasks) that include sprint planning, daily stand-ups, and sprint reviews. It should be noted that Scrum dominates the Agile space with a large 74 percent of the market share when including the Scrum/XP hybrid approach. For this reason, many people often use the terms Agile and Scrum somewhat synonymously even though their definitions are different.

In contrast to XP, Scrum is more focused on the planning aspects of Agile development rather than on the technical processes. This includes requirements management as well as iteration and release planning. As a result, Scrum can be used as a management framework used with additional Agile technical methodologies. The following information on Scrum roles and processes is derived from The Scrum Primer.26

SCRUM ROLES
- The product owner identifies and prioritizes required features and is responsible for the overall project success.  
- The team builds the product based on direction from the product owner. The team is cross-functional (functional analyst, developer, tester,
database designer, etc.) and is typically no larger than 10-15 people.

- The scrum master helps the team and product owner follow the Scrum process by removing barriers to the team’s progress, resolving issues, and enabling the team to be successful.

**SCRUM PROCESS**

1. The process begins with the product owner collecting and prioritizing requirements in a product backlog.
2. The team and product owner work together to select the highest priority requirements that can be accomplished within a given time period called a sprint.
3. The scrum master facilitates the team through daily collaborative meetings, buffers them from external interference, and removes barriers that slow their progress.
4. At the end of the sprint, the team demonstrates the completed features comprising a potential shippable product to the product owner. The feedback gathered from the demonstration is fed back into the product backlog.
5. The team then has a retrospective to determine what worked well and what improvements need to be made to the process for the next sprint.

**LEAN SOFTWARE DEVELOPMENT** was described by Mary and Tom Poppendieck and is based on Toyota manufacturing principles: eliminate waste, amplify learning, decide as late as possible, deliver as fast as possible, empower the team, build integrity in, and see the whole. Lean manufacturing has been in use for more than 50 years, and includes processes such as just-in-time production. Lean software development, a more recent concept, provides a set of overarching principles that can be used to help guide decision making during the development process. These are related to goals such as making quality an integral part of the process and removing waste by minimizing work in progress and reducing excessive documentation. As a result, Lean software development can be used with any combination of Agile methods to optimize the development process. The 10 basic practices or rules of Lean production are:

1. **Eliminate waste.** Reduce things that do not produce value such as excess documentation, functionality that is not used, and waiting time between processes (such as review or approval).
2. **Minimize inventory.** For software development, examples are excess documentation that will not be released with the code and code that is not finished.
3. **Maximize flow.** Reduce the time it takes to deliver value, such as smaller release cycles.
4. **Pull from demand.** Respond to change instead of predicting needs.
of trying to predict it. Keep requirements flexible and make decisions at the last possible moment.

5. Empower workers. Push decisions down to the lowest level possible. To do this, the team must understand the project vision and have the data and authority to make decisions.

6. Meet customer requirements. Team with the customer throughout development rather than at the beginning, when requirements are defined, and the end when they are verified.

7. Do it right the first time. Make quality and feedback intrinsic aspects of the development workflow rather than as separate processes.

8. Ban local optimization. Do not sub-optimize requirements by locking them down at the beginning of the project. Things will change. The customer’s needs will change.

9. Partner with suppliers. Build relationships on trust and doing what is best for the customer, rather than highly specified contracts. These contracts can result in adversarial relationships.

10. Create a culture of continuous improvement. Encourage learning and feedback to continually improve the processes.

FEATURE-DRIVEN DEVELOPMENT (FDD) is a model-driven, iterative process in which five activities are performed: develop overall model, build feature list, plan by feature, design by feature, and build by feature. Six milestones are defined for each feature.

AGILE UNIFIED PROCESS (AUP) is an Agile version of the Rational Unified Process developed by Scott Ambler. It uses a number of Agile techniques in the context of eight disciplines: model, implementation, test, deployment, configuration management, project management, and environment.30

CRYSTAL METHODS emphasize “peopleware” issues rather than process. They were developed by the well-known Agile theorist Alistair Cockburn who defines software development as “… a cooperative game of invention and communication.” Cockburn defines a number of Crystal methods with varying amounts of documentation and process based on staff size, criticality, and project priority.31

DYNAMIC SOLUTIONS DELIVERY MODEL (DSDM) fixes cost, quality, and time at the outset of the project, and then uses MoSCoW (must have, should have, could have, and won’t have) prioritization to work within the time constraint.

STRENGTHS AND WEAKNESSES

Each methodology has its own unique strengths and weaknesses that make it more appropriate in certain contexts. And in many cases core concepts can be combined, e.g., Scrum and Lean fit well together. Scrum is a management approach often combined with XP. See table 2.


### Table 2 Comparison of Methodologies – Strengths, Weaknesses, and Uniquenesses

<table>
<thead>
<tr>
<th>AGILE METHODOLOGY</th>
<th>STRENGTHS</th>
<th>WEAKNESSES</th>
<th>UNIQUENESS</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>XP</strong></td>
<td>• Strong technical practices.&lt;br&gt;• Customer ownership of feature priority, developer ownership of estimates.&lt;br&gt;• Frequent feedback opportunities.&lt;br&gt;• Most widely known and adopted approach, at least in the U.S.</td>
<td>• Requires on-site customer.&lt;br&gt;• Documentation primarily through verbal communication and code. For some teams these are the only artifacts created; others create minimal design and user documentation.&lt;br&gt;• Difficult for new adopters to determine how to accommodate architectural and design concerns.</td>
<td>• Pair programming&lt;br&gt;• 10-minute build&lt;br&gt;• Slack&lt;br&gt;• 10-12 team members&lt;br&gt;• TDD&lt;br&gt;• First-test programming</td>
</tr>
<tr>
<td><strong>SCRUM</strong></td>
<td>• Complements existing practices,&lt;br&gt;• Self-organizing teams and feedback.&lt;br&gt;• Customer participation and steering.&lt;br&gt;• Priorities based on business value.&lt;br&gt;• Only approach here that has a certification process.</td>
<td>• Only provides project management support, other disciplines are out of scope.&lt;br&gt;• Does not specify technical practices.&lt;br&gt;• Can take some time to get the business to provide unique priorities for each requirement.</td>
<td>• PM wrapper&lt;br&gt;• Commitment to defined goal&lt;br&gt;• Sprint planning meeting&lt;br&gt;• 30-day sprints maximum&lt;br&gt;• Team size 7 +/-2&lt;br&gt;• Story driven&lt;br&gt;• Customer determined</td>
</tr>
<tr>
<td><strong>LEAN</strong></td>
<td>• Complements existing practices.&lt;br&gt;• Focusses on project ROI.&lt;br&gt;• Eliminates all project waste.&lt;br&gt;• Uses cross-functional teams.</td>
<td>• Does not specify technical practices.&lt;br&gt;• Requires constant gathering of metrics, which may be difficult for some environments to accommodate.&lt;br&gt;• Theory of Constraints can be a complex and difficult aspect to adopt.</td>
<td>• Can be used with Scrum as an underlying philosophy.&lt;br&gt;• Supports many Scrum methods.&lt;br&gt;• Focus on environmental whole of work performance.</td>
</tr>
<tr>
<td><strong>FDD</strong></td>
<td>• Supports multiple teams working in parallel.&lt;br&gt;• All aspects of a project tracked by feature.&lt;br&gt;• Design by feature and build by feature aspects are easy to understand and adopt.&lt;br&gt;• Scales to large teams or projects well.</td>
<td>• Promotes individual code ownership as opposed to shared/team ownership.&lt;br&gt;• Iterations are not as well defined by the process as other Agile methodologies.&lt;br&gt;• The model-centric aspects can have huge impacts when working on existing systems that have no models.</td>
<td>• Highly specified development practices.&lt;br&gt;• High-level object-model diagrams.&lt;br&gt;• Management-directed team efforts.&lt;br&gt;• Very structured 5-step process.&lt;br&gt;• Scalable to large teams.&lt;br&gt;• Originally developed for 15-month, 50-person development project.</td>
</tr>
<tr>
<td><strong>AUP</strong></td>
<td>• Robust methodology with many artifacts and disciplines to choose from.&lt;br&gt;• Scales up very well.&lt;br&gt;• Documentation helps communicate in distributed environments.&lt;br&gt;• Priorities set based on highest risk. Risk can be a business or technical risk.</td>
<td>• Higher levels of ceremony may be a hindrance in smaller projects.&lt;br&gt;• Minimal attention to team dynamics.&lt;br&gt;• Documentation is much more formal than most approaches mentioned here.</td>
<td>• Follows 5 principles of RUP:&lt;br&gt;• Model&lt;br&gt;• Implementation&lt;br&gt;• Test deployment&lt;br&gt;• Configuration management&lt;br&gt;• Project management&lt;br&gt;• Environment&lt;br&gt;• Philosophy:&lt;br&gt;• Your staff knows what they’re doing&lt;br&gt;• Simplicity&lt;br&gt;• Agility&lt;br&gt;• Focus on high-value activities&lt;br&gt;• Tool independence&lt;br&gt;• The AUP to meet your own needs.</td>
</tr>
<tr>
<td>AGILE METHODOLOGY</td>
<td>STRENGTHS</td>
<td>WEAKNESSES</td>
<td>UNIQUENESS</td>
</tr>
<tr>
<td>-------------------</td>
<td>------------</td>
<td>------------</td>
<td>------------</td>
</tr>
</tbody>
</table>
| Crystal           | - Family of methodologies designed to scale by project size and criticality.  
                    - Only methodology that specifically accounts for life-critical projects.  
                    - As project size grows, cross-functional teams are used to ensure consistency.  
                    - The “human” component has been considered for every aspect of the project support structure.  
                    - An emphasis on testing is so strong that at least one tester is expected to be on each project team. | - Expects all team members to be collocated. May not work well for distributed teams.  
                    - Adjustments are required from one project size/structure to another to follow the prescribed flavor of Crystal for that project size/criticality.  
                    - Moving from one flavor of Crystal to another in mid-project doesn’t work, as Crystal was not designed to be upward or downward compatible. | - 4-month duration cycle maximum; working deliveries ideally every 1-2 months.  
                    - Clear and Yellow are well-defined.  
                    - Method adopted depends on product criticality.  
                    - Clear is for collocated teams of 8 or fewer.  
                    - Yellow is for teams of 10-20 people.  
                    - Orange is for 20-50 people, Red is for 50-100 people and so on, through Maroon, Blue and Violet.  
                    - Twofold focus: Deliver system and set up for new game.  
                    - Unit tests designed by designer-programmer.  
                    - Automated regression tests. |
| DSDM              | - An emphasis on testing is so strong that at least one tester is expected to be on each project team.  
                    - Designed from the ground up by business people, so business value is identified and expected to be the highest priority deliverable.  
                    - Has specific approach to determining how important each requirement is to an iteration.  
                    - Sets stakeholder expectations from the start of the project that not all requirements will make it into the final deliverable. | - Probably the most heavyweight project compared in this survey.  
                    - Expects continuous user involvement.  
                    - Defines several artifacts and work products for each phase of the project; heavier documentation.  
                    - Access to material is controlled by a Consortium. | - Focus on the business need.  
                    - Deliver on time.  
                    - Collaborate.  
                    - Never compromise quality.  
                    - Build incrementally from firm foundations.  
                    - Develop iteratively.  
                    - Communicate continuously and clearly.  
                    - Demonstrate control.  
                    - Perform an objective evaluation to ensure quality, including finding defects, validating that the system works as designed, and verifying that the requirements are met. |
## Table 3 Comparison of Methodologies – Roles

<table>
<thead>
<tr>
<th>AGILE METHODOLOGY</th>
<th>ROLES</th>
</tr>
</thead>
<tbody>
<tr>
<td>XP</td>
<td>Manager</td>
</tr>
<tr>
<td></td>
<td>Scrum master</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>SCRUN</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>LEAN</td>
<td>· Executive leadership</td>
</tr>
<tr>
<td></td>
<td>· Line management</td>
</tr>
<tr>
<td></td>
<td>· Program manager</td>
</tr>
<tr>
<td>FDD</td>
<td>Project manager</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>AUP</td>
<td>· Project manager</td>
</tr>
<tr>
<td></td>
<td>· Configuration manager</td>
</tr>
<tr>
<td></td>
<td>· Change control manager</td>
</tr>
<tr>
<td></td>
<td>· Deployment manager</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>CRYSTAL</td>
<td>Project manager</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>DSDM</td>
<td>Project manager</td>
</tr>
<tr>
<td></td>
<td>Technical coordinator</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>
**MANAGEMENT TOOL REVIEW**

<table>
<thead>
<tr>
<th>PM</th>
<th>PRACTICES</th>
<th>TEAMPULSE</th>
<th>GREENHOPPER</th>
<th>LEANKIT KANBAN</th>
<th>TARGET PROCESS</th>
<th>MINGLE</th>
<th>VERSIONONE</th>
<th>ONTIME-SCRUN</th>
<th>PIVOTAL TRACKER</th>
<th>RALLY</th>
<th>SCRUMWISE</th>
<th>LITHESPEED</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>SCRUM CONCEPTS</strong></td>
<td>Backlog/Work Items</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td></td>
<td>Release Planning</td>
<td>X</td>
<td>X</td>
<td></td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Task Boards</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Sprint Planning</td>
<td>X</td>
<td>X</td>
<td></td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td></td>
<td>Burndown Reports</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Sprint Retrospectives</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Dashboards</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>LEAN CONCEPTS</strong></td>
<td>Kanban Boards</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Team Boards</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Flow Optimization</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Flow Reports</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>ADDITIONAL FEATURES</strong></td>
<td>Collaboration Features</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Workflow Customization</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Product Roadmap Capability</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Project Portfolio Management</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Time Tracking</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td></td>
<td>Import/Export</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Integrations</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Mobile App Option</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Bug Tracking</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>JIRA</td>
<td>PLANBOX</td>
<td>PM ROBOT</td>
<td>AGILE FANT</td>
<td>PANGOSCRUM</td>
<td>ICE SCRUM</td>
<td>SCRUM/HALF</td>
<td>BRIGHT GREEN</td>
<td>SCRUMPAD</td>
<td>SCRUMWORKS PRO</td>
<td>SCRUMdesk</td>
<td>AGILE FOR SCRUM</td>
<td>TINYPM</td>
</tr>
<tr>
<td>------</td>
<td>---------</td>
<td>----------</td>
<td>------------</td>
<td>-------------</td>
<td>-----------</td>
<td>------------</td>
<td>--------------</td>
<td>----------</td>
<td>----------------</td>
<td>-----------</td>
<td>----------------</td>
<td>------</td>
</tr>
<tr>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td></td>
</tr>
<tr>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td></td>
</tr>
<tr>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td></td>
</tr>
<tr>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td></td>
</tr>
<tr>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td></td>
</tr>
<tr>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td></td>
</tr>
<tr>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td></td>
</tr>
<tr>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td></td>
</tr>
<tr>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td></td>
</tr>
<tr>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td></td>
</tr>
<tr>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td></td>
</tr>
<tr>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td></td>
</tr>
<tr>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td></td>
</tr>
<tr>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td></td>
</tr>
</tbody>
</table>
AGILE TOOL SNAPSHOT
ATLASSIAN | GREENHOPPER

INTEGRATION
GreenHopper allows scrum teams to visualize workflows by providing horizontal swim lanes. It permits planning teams to limit work in progress (WIP), measure cycle times, and conduct trend analysis.

SCOPE
GreenHopper provides scrum teams backlog grooming for estimating stories and anticipating sprint planning.

GreenHopper provides teams the ability to decompose stories into tasks and include acceptance criteria to clarify the collective definition of “done.”

Through “Sprint Marker” implementation, sprint teams commit to predefined sprint backlog at the outset of the sprint.

Through its Sprint Report, GreenHopper describes the initial sprint commitment, related scope changes, and sprint team delivery.

TIME
GreenHopper incorporates JIRA Project Tracking to plan, track, and analyze project schedules. The Sprint Marker enables the product owner to schedule specific stories from the product backlog to the sprint backlog. Through this use of JIRA, scrum teams manage issues, tasks, bugs, deadlines, code, and hours/burn.

COMMUNICATION
Using the GreenHopper Confluence Collaboration tool, scrum teams create and share documents, files, ideas, specs, diagrams, mockups, and calendars. Additionally, teams use this tool to track projects, events, and resources.

Through story swim lanes, scrum teams view story progress, and product owners accept and resolve completed stories.

Configurable JIRA wallboards provide an information radiator teams use at daily standup along with sprint statistics, a burndown chart, sprint days remaining, and cumulative flow diagrams.

PRODUCT COST
$10 per month per 10 users to $500 per month per 2,000 users
**AGILE TOOL SNAPSHOT**
**AXOSOFT | ONTIME SCRUM**

**INTEGRATION**

OnTime Scrum provides “at a glance” real-time project visibility, enabling teams to measure progress and make decisions.

Customized dashboards facilitate tailoring by product, release, or combining multiple projects into single views.

**SCOPE**

OnTime Scrum provides scope planning through its “OnTime Card.” These kanban or scrum planning boards allow the project management team to plan and view backlogs, user stories, bug tracking, and workflow.

**TIME**

OnTime Scrum provides scheduling capability to plan versions, releases, and sprints and maintain backlog ordering with drag and drop stack ranking.

Progressing from one release to the next, OnTime Scrum automatically updates burn down charts to reflect the current data, enabling project management.

**COMMUNICATION**

The OnTime Card dashboard provides real-time, interactive planning projectable to working space screens. Through this dashboard, the scrum team views team assignment, priority, release and worklog information. The team can also indicate priority, status, severity, or workflow steps.

**PRODUCT COST**

$7.00 per month per user or $100 per year for teams of up to 10 users
AGILE TOOL SNAPSHOT
LEANKIT | LEANKIT KANBAN

INTEGRATION

LeanKit is cloud-hosted so there’s no software to install. As a web browser-based system, LeanKit is available from anywhere and anytime there is an Internet connection.

LeanKit boards can be designed to match even the most complex process for any industry with no programming. The company offers an expanding library of board templates. And you can save your own templates to spread best practice standards within your organization. Integrations can be written to make LeanKit a visual front-end to other systems like your customer support ticketing system, ERP, or CRM.

SCOPE

Online, Visual Project Boards: On each board the process steps are represented as vertical and horizontal lanes. Cards represent work items, which team members update and move from across the board as they complete their share of the work. Rather than having to ask for status reports, managers and customers can just look at the board. Board updates are visible in seconds around the globe and e-mail alerts and RSS feeds are available.

Project Status: Different card colors can be set to represent different types of work, like deliverables, issues, and risks. Icons on the cards indicate who’s assigned, their priority, due dates, relative sizes, and problems.

Analytics: The history of each card is automatically recorded and kept forever. This history also drives analytical reports. LeanKit historical data can also be easily exported and combined with other company data for custom reporting.

TIME

With LeanKit project and portfolio management, an organization’s work can be broken down into multiple levels of detail for different audiences. Those can then be delegated to the actual teams who will do the work. A team’s board might contain deliverables delegated to them from several different project boards plus routine non-project work. You can create taskboards inside each card to keep track of the smaller checklist items required to complete it. At each level, users can view snapshot metrics for cards they have delegated to another board or taskboard items, letting them quickly see how well the work is progressing.

COMMUNICATION

Leaders can get the information they need for accurate forecasting and targeted process improvement. The history of each card is automatically recorded and kept forever. This provides an easy audit trail of who did what and when. This history also drives analytical reports. LeanKit historical data can also be easily exported and combined with other company data for custom reporting.

PRODUCT COST

Free to $15 per user per month

LeanKit, Inc.
251 2nd Ave. South, Ste. 205
Franklin, TN 37064

O: 615-219-7424

http://leankit.com/
AGILE TOOL SNAPSHOT

PIVOTAL LABS, INC. | PIVOTAL TRACKER

INTEGRATION

Pivotal Tracker translates, tracks, and decomposes project features into manageable user stories. It includes estimating complexity, labeling and linking like stories, applying external files or mock-ups, prioritizing stories into release, then sprint backlogs, and then into sprint execution. It integrates with standard off-the-shelf development and tracking tools.

SCOPE

Velocity and emergent iterations: Tracker calculates future iterations automatically, based on historical performance.

TIME

Pivotal Tracker provides for planning releases and focusing on milestones. It adapts as real-world changes occur.

Pivotal Tracker calculates completion dates automatically, based on velocity. Estimations adjust in real time as the team adds new stories or reprioritizes, via drag and drop, to reflect evolving priorities.

COMMUNICATION

Pivotal Tracker communicates project status through a single shared view that facilitates team communication of near-term activities and future efforts. Tracker embeds charts, including release burndown, iteration burnup, story type breakdown, and historical velocity. It permits reordering project stories to reflect evolving priorities. Through intuitive drag/drop and real-time updating, scrum teams can immediately see the impacts of changes to release schedules. Additional access to stories and status is provided through Tracker’s full text and structured query search.

PRODUCT COST

Free to $175 per month for up to 50 users

Pivotal Labs HQ - USA
875 Howard St., 5th Floor
San Francisco, CA 94103
415.77.PIVOT
contact@pivotallabs.com
www.pivotaltracker.com
AGILE TOOL SNAPSHOT
RALLY | RALLY

INTEGRATION
Rally Software provides a single central structure from which scrum teams can build and rank backlogs, estimate size, and groom items for planning and scheduling.

Rally scales, by product line, to support Agile development by a single team to multiple teams.

Rally editions align customer organizational structure.

Rally facilitates coordination among multiple projects and aggregation of program level roadmaps, rolling up features, dependencies, and blocking issues from individual projects into a single program status view.

SCOPE
Rally software enables scrum teams to plan out releases and associated content by capturing, refining, ranking, and tracking user ideas from inception through delivery.

Scrum teams develop product roadmaps, decompose requirements, and maintain traceability from strategic plans to deliverables.

Rally aligns stories and coordinates multiple release backlogs to ensure all scrum teams can trace sprint efforts to higher-level features and organization release commitments.

TIME
Rally software provides a single source for teams to log hours and costs and managers to track these for burn, budgets, and billing.

COMMUNICATION
Use Rally’s reporting capabilities to track the progress of features, projects or portfolios. Use our point-and-click Report Designer to create custom reports from Rally’s SaaS-based ALM analytics warehouse.

Keep multi-team releases in sync with reports and views that automatically roll up progress against shared commitments.

Centralize everything the team needs to build and rank backlogs, estimate size, and groom items for planning and scheduling.

Participate in the flow of activities on your team and tailor your views and notifications to help you get the right things done. Create your own custom dashboards and share them with your team.

Give your entire organization real-time visibility into the status of features, quality, priorities, roadblocks and risks. Contribute to team discussions and instantly communicate with team members—without leaving the application.

PRODUCT COST
Free to $49.00 per user per month

Rally Software
3333 Walnut St.
Boulder, CO 80301

O: 866.348.1552
F: 303.226.1179

info@rallydev.com
www.rallydev.com
AGILE TOOL SNAPSHOT
TARGETPROCESS, INC. | TARGETPROCESS

INTEGRATION
TargetProcess Release Plan enables scrum teams to integrate across the current release and future release planning by creating high-level plans for several projects on one screen.

Teams create an integrated roadmap to plan multiple releases for multiple projects, set milestones, and control the scope of current and future releases.

SCOPE
TargetProcess Sprint Plan enables scrum teams to quickly plan sprints, control velocity, prioritize user stories, and resolve bugs.

TargetProcess single screen Backlog enables scrum teams to refine features, user stories, and bugs in a single backlog. Through Backlog, the product owner adds, updates, prioritizes, splits, merges, and categorizes user stories.

TIME
TargetProcess Release Plan provides a cross-project roadmap in a single-screen display. Scrum teams use this roadmap to schedule multiple releases across multiple projects, set milestones, and control the scope of future releases. Release Plan enables teams to track progress for the current release as well.

COMMUNICATION
Scrum teams communicate status through the Task Board, Teams Board, and Burndown Chart Board. Through the Task Board, teams track progress, assign work, run daily meetings, and update tasks. Through the Teams Board, teams reassign people, move stories to other teams and projects, run daily meetings, and view collective progress. Through the Burndown Chart, teams view changes to releases and sprints and forecast release and sprint end dates based on the history data.

PRODUCT COST
Free to $25 per user per month
AGILE TOOL SNAPSHOT
TEGRICK | TEAMPULSE

INTEGRATION
TeamPulse provides real-time project status through its TeamPulse Project Dashboard. The dashboard displays a burndown chart, blocking issues, scope changes, and work completion forecast.

It also provides two virtual boards—a story board and task board that teams can use to create or edit work items, filter by people, and visualize iterations.

Finally, it manages risks and issues.

SCOPE
TeamPulse provides an intuitive user interface for effective planning and scheduling of releases and sprints. It constrains capacity and migrates scope through the backlogs, from product to release to sprint.

TeamPulse facilitates defining and managing of Agile personas. Agile teams use personas to describe end-user interaction through the use of modeling stories.

TeamPulse organizes project backlog items (e.g., bugs, issues, risks) in one place, to facilitate oversight (i.e., view, edit, relate, and prioritize them).

TIME
TeamPulse facilitates scheduling of release plans, establishing release start and end dates, separating releases into shorter iterations with finite duration and capacities, and networking through predecessor/successor relationships.

By enabling the work-in-progress (WIP) limits, teams use TeamPulse boards as kanban boards to improve the flow of work and do scrum.

PRODUCT COST
$1,499 per five users to $2,999 for unlimited users
AGILE TOOL SNAPSHOT
THOUGHTWORKS STUDIOS | MINGLE

INTEGRATION
Mingle facilitates scrum planning by providing real-time information necessary to guide project delivery.

Through dashboards, taskboards, and configurable card walls, scrum teams capture and visualize team activity, track release and sprint status, and monitor and manage communications, scope changes, and critical issues.

SCOPE
Mingle facilitates easy project requirements capture. Scrum teams manage features, epics, and stories using searchable, filterable, multidimensional card walls and tree hierarchies.

Program backlogs enable teams to manage and collaborate on future high-level work as well as forecast based on project plans.

TIME
Mingle supports planning and replanning as business needs evolve. Teams create visual, interactive timelines and receive alerts if projects begin to slip.

Providing real-time information and iterative feedback, Mingle permits teams to schedule releases, iterations, or sprints and prioritize work using drag and drop card ranking. Mingle makes it easy to understand exactly how changes affect delivery timelines and adapt accordingly.

COMMUNICATION
Mingle provides customizable metrics and reporting containing real-time information. Dashboards allow teams to refine and drill down into details as needed. Customizable reports and dashboards display burndowns, burnups, and other Agile metrics. Data filters limit information at specific levels, and on-the-fly reports, burn-up and burn-down charts, and velocity metrics provide teams insight across projects and at selected levels within projects.

PRODUCT COST
$240 per user per year to $400 per user per month
AGILE TOOL SNAPSHOT
VERSIONONE | VERSIONONE

INTEGRATION
VersionOne provides product planning through its integrated roadmapping, allowing scrum teams to visualize and manage product requirements, epics, stories, and goals across projects, teams, and the portfolio.

Scrum teams can align product timelines with corporate objectives, collaborate to optimize software deliveries, and integrate teams and stakeholders across the enterprise.

SCOPE
VersionOne incorporates an intuitive, drag-and-drop release backlog environment to facilitate scrum team release planning, forecasting, and analysis.

VersionOne incorporates an intuitive sprint backlog to facilitate iterative sprint planning of stories, defects, tasks, tests, and impediments.

TIME
VersionOne allows development of product release strategies using integrated roadmapping capabilities. Scrum teams create timelines of related projects and release milestones to ensure deliverables align vertically from corporate objectives to product deliverables. Timelines can be adjusted as the business environment changes.

COMMUNICATION
VersionOne incorporates sprint tracking through interactive storyboards, taskboards, testboards, and burndown charts.

Sprint review quickly closes out sprints and capture relevant issues and action items from your team’s retrospectives in a single location.

PRODUCT COST
Free to $39 per user per month
### APPENDIX A: AGILE GLOSSARY OF TERMS

<table>
<thead>
<tr>
<th>TERM</th>
<th>DEFINITION</th>
</tr>
</thead>
<tbody>
<tr>
<td>40-HOUR WORK WEEK</td>
<td>The number of hours for a sustainable workload</td>
</tr>
<tr>
<td>AGILE</td>
<td>The basis for several software development methodologies, a philosophy emphasizing simplicity, customer satisfaction through early and continuous delivery, face-to-face conversation, and short iterations</td>
</tr>
<tr>
<td>AGILE MANIFESTO</td>
<td>A series of principles and values for software development or project/program execution</td>
</tr>
<tr>
<td>BACKLOG</td>
<td>A prioritized list of stories or requirements to be worked on; may include user stories that, due to time or other resource constraints, could not be addressed in a given sprint</td>
</tr>
<tr>
<td>BURNDOWN CHART</td>
<td>A line chart representing the measurement of progress for an Agile project at both an iteration and a project level; progress is shown by line down from the Y to X axis</td>
</tr>
<tr>
<td>BURNUP CHART</td>
<td>A line chart representing the measurement of progress for an Agile project at both an iteration and a project level; progress is shown by line up from the X to Y axis</td>
</tr>
<tr>
<td>CHICKENS</td>
<td>A term for interested stakeholders, managers, and executives who serve as observers of the project or process and may consult as needed but are not accountable for specific outcomes of the project or process</td>
</tr>
<tr>
<td>CHICKENS AND PIGS</td>
<td>The principal players in an anecdotal story analogous to the real-life experiences of individuals participating in an Agile project</td>
</tr>
<tr>
<td>COLLECTIVE CODE OWNERSHIP</td>
<td>Accountability of more than one programmer for maintaining code</td>
</tr>
<tr>
<td>CUSTOMER COLLABORATION</td>
<td>Engagement of external stakeholders, such as product users, to deliver a product that meets the customer's requirements</td>
</tr>
<tr>
<td>CYCLE TIME</td>
<td>The time required to complete a cycle of an iteration</td>
</tr>
<tr>
<td>DAILY SCRUM/STANDUP</td>
<td>A brief meeting that usually lasts 15 minutes or less held at the same time each day whereby each team member informs the group of his/her previous day's activities, any planned activities for the current day, and any impediments</td>
</tr>
<tr>
<td>DEVELOPMENT TEAM</td>
<td>A cross-functional group of people responsible for delivering potentially deployable increments of a product at the end of every sprint</td>
</tr>
<tr>
<td>DONE (ALSO REFERRED TO AS “DONE DONE” AND “DEFINITION OF DONE”)</td>
<td>Exit criteria that determines the completion all the various tasks</td>
</tr>
<tr>
<td>EMERGENCE</td>
<td>Continual delivery of functionality with business value as it is needed</td>
</tr>
<tr>
<td>EMPIRICISM</td>
<td>A contextual approach whereby teams or individuals try something out, learn from it, and then reflect on the experience and change if necessary; also known as the principle of inspect and adapt</td>
</tr>
<tr>
<td>TERM</td>
<td>DEFINITION</td>
</tr>
<tr>
<td>-----------------------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>EPIC</td>
<td>A very large user story, or a group of related stories, used at the program and project levels in product roadmaps and backlog for features not yet sufficiently analyzed to break into component stories</td>
</tr>
<tr>
<td>ESTIMATION</td>
<td>A process that involves collaboration and agreement on a size measurement for the stories as well as the tasks required to implement those stories in a product backlog</td>
</tr>
<tr>
<td>EVOLVING REQUIREMENTS</td>
<td>Requirements (i.e., user stories) based on emerging developments</td>
</tr>
<tr>
<td>FLOW</td>
<td>Continuous delivery of value to customers</td>
</tr>
<tr>
<td>FIBONACCI SEQUENCE</td>
<td>The sequence of numbers where the next number is derived by adding together the previous two numbers, the sequence is often used for story points, simply because estimates are always less accurate when dealing with epics</td>
</tr>
<tr>
<td>IMPEDIMENT</td>
<td>Anything that prevents the team from performing work and requires, if it is organizational in nature, that the scrum master eliminate</td>
</tr>
<tr>
<td>IMPEDIMENT BACKLOG</td>
<td>A list of everything preventing a team member from performing work or blocking the team from being productive</td>
</tr>
<tr>
<td>ITERATION</td>
<td>A two- to four-week period devoted to delivering an increment of useful business functionality</td>
</tr>
<tr>
<td>JUST IN TIME</td>
<td>The production and delivery of an item, such as a user story or documentation, to the right place at the right time</td>
</tr>
<tr>
<td>KAIZEN</td>
<td>Continuous incremental improvement</td>
</tr>
<tr>
<td>KANBAN</td>
<td>A method that emphasizes just-in-time delivery and the optimization of flow of work for the team</td>
</tr>
<tr>
<td>LEAN</td>
<td>A set of techniques and principles for delivering more value with fewer resources; underlying concept is that the expenditure of resources on anything other than the creation of value for the end customer is wasteful and should be eliminated</td>
</tr>
<tr>
<td>META-SCRUM</td>
<td>The coordination of resources and dependencies to accomplish completion of a product or initiative through a mega-meeting of scrum team members acting as “ambassadors” who participate in a daily meeting with ambassadors from other teams as well as product owners, stakeholders, business owners, and sponsors; also known as a “scrum of scrums”</td>
</tr>
<tr>
<td>PAIR PROGRAMMING</td>
<td>A technique in which two programmers work in tandem at one workstation, with one typing in code while the other reviews each line of code as it is typed in</td>
</tr>
<tr>
<td>PIGS</td>
<td>A term used to describe committed individuals directly accountable for specific project/product deliverables They participate wholly in the process, and their involvement and estimates set expectations to which they will be held accountable</td>
</tr>
<tr>
<td>TERM</td>
<td>DEFINITION</td>
</tr>
<tr>
<td>------</td>
<td>------------</td>
</tr>
<tr>
<td>PLANNING POKER (ALSO CALLED SCRUM POKER)</td>
<td>A consensus-based technique for estimating level of effort and relative size of tasks</td>
</tr>
<tr>
<td>POINT SCALE/EFFORT/STORY POINTS</td>
<td>A subjective numerical system for assigning a story's difficulty level</td>
</tr>
<tr>
<td>PRODUCT BACKLOG</td>
<td>A repository for high-level requirements with high level-estimates, typically provided by the product stakeholders, targeted for release at some point</td>
</tr>
<tr>
<td>PRODUCT CHARTER</td>
<td>A definition of the product's purpose for existing, what will be done to achieve the product's purpose, and success criteria</td>
</tr>
<tr>
<td>PRODUCT OWNER</td>
<td>An individual accountable for ensuring the team delivers a product based on inputs from various stakeholders</td>
</tr>
<tr>
<td>PULL</td>
<td>Stored items in the queue waiting to be produced</td>
</tr>
<tr>
<td>PUSH</td>
<td>Items produced according to schedule, often ordered and produced before required and stored in a batch and queue system until the upstream operator is ready</td>
</tr>
<tr>
<td>REFACTORIZING</td>
<td>A technique for altering the internal structure of a body of code without changing its intended behavior</td>
</tr>
<tr>
<td>RELEASE BURNDOWN/BURNUP CHART</td>
<td>A visible chart that shows a sprint-level progress toward a release or story in the product backlog; see distinction between burndown/burnup charts above</td>
</tr>
<tr>
<td>RELEASE PLAN</td>
<td>A plan for implementing a release, which includes a list of all anticipated activities, resources, and responsibilities as well as an estimated duration of the sprint that will lead to the release</td>
</tr>
<tr>
<td>RETROSPECTIVE</td>
<td>A team meeting occurring at the end of every sprint/iteration to review lessons learned and discuss how the team can be more efficient in the future</td>
</tr>
<tr>
<td>ROADMAP</td>
<td>The vision of a product (from product charter) explained in a high-level plan that outlines project work spanning one or more releases, and groups requirements into prioritized themes before estimating an execution schedule against themes</td>
</tr>
<tr>
<td>SCHEDULE-BASED PLANNING</td>
<td>Planning driven by schedule and completion/release dates rather than features and scope, with release plans based on estimates of scope that can be completed in a realistically defined release time box</td>
</tr>
<tr>
<td>SCRUM</td>
<td>Based on the adaptive and iterative methodology of software development, a framework for addressing complex adaptive problems while productively and creatively delivering products of the highest possible value</td>
</tr>
<tr>
<td>SCRUM MASTER</td>
<td>An individual who acts as a buffer between the team and any distracting influences while ensuring the scrum process is used as intended and enforcing the rules</td>
</tr>
<tr>
<td>SCRUM OF SCRUMS</td>
<td>A meeting of multiple scrum teams, with each team’s scrum master as a designated representative, gathering to discuss management of work that overlaps among the teams and any interdependencies as well as provide a status on team accomplishments</td>
</tr>
<tr>
<td>TERM</td>
<td>DEFINITION</td>
</tr>
<tr>
<td>-----------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>SELF-ORGANIZATION</td>
<td>The ability of those closest to the work to know how best to accomplish it, set clear goals and boundaries, and make all tactical and implementation decisions</td>
</tr>
<tr>
<td>SIMPLE DESIGN</td>
<td>A design that passes its tests, minimizes duplication, maximizes clarity, and has as few elements as possible without sacrificing the objectives of design</td>
</tr>
<tr>
<td>SPIKE</td>
<td>A time-boxed piece for researching a concept or creating a simple prototype before delivery of large epics or user stories to help steer decisions on a course of action</td>
</tr>
<tr>
<td>SPRINT/ITERATION</td>
<td>A fixed duration, typically between 2- to 4-week increments, of software development activities that lead to producing a product</td>
</tr>
<tr>
<td>SPRINT BACKLOG</td>
<td>User stories that have not yet been addressed, but which are candidates to be addressed in future sprints</td>
</tr>
<tr>
<td>SPRINT EXECUTION</td>
<td>Also referred to as an iteration execution, a recurring phase of a project life cycle in which elements of the iteration backlog are executed with the goal being to deliver a usable product (element) within time constraints of the iteration, addressing as many commitments as possible while not compromising quality or schedule</td>
</tr>
<tr>
<td>SPRINT PLAN</td>
<td>Also referred to as an iteration plan, a detailed execution plan for an iteration (usually the current one) that defines the goals and commitments and specifies the requirements, priorities, and team member work assignments required to complete the sprint</td>
</tr>
<tr>
<td>SPRINT PLANNING MEETING</td>
<td>A meeting of the core sprint team whereby the product owner describes the highest priority features indicated on the product backlog; the team agrees on the number of features to accomplish and plans out the tasks required to achieve delivery of those features</td>
</tr>
<tr>
<td>SPRINT RETROSPECTIVE</td>
<td>A three-hour reflection and discussion, facilitated by the scrum master, on lessons learned from completing a sprint</td>
</tr>
<tr>
<td>SPRINT REVIEW</td>
<td>A four-hour review of the complete or incomplete software or work with the stakeholders</td>
</tr>
<tr>
<td>STAKEHOLDER</td>
<td>Anyone with an interest or affected by the product or project</td>
</tr>
<tr>
<td>STORY POINTS</td>
<td>See point scale/effort above</td>
</tr>
<tr>
<td>TASK</td>
<td>An activity that begins at the start of the sprint and lasts no longer than 12 hours; it also a component of a user story</td>
</tr>
<tr>
<td>TASKBOARD/STORYBOARD</td>
<td>A display of cards and sticky notes on a wall chart to represent all the work, in the form of user stories, in a given sprint</td>
</tr>
<tr>
<td>TEAM</td>
<td>Five to nine people, with cross-functional skills, who are self-organizing and self-led, but often work with some form of project or team management to deliver a product</td>
</tr>
<tr>
<td>TERM</td>
<td>DEFINITION</td>
</tr>
<tr>
<td>-----------------------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>TEAM MEMBER</td>
<td>Anyone, except for the product owner or scrum master, on a scrum team responsible for a specific sprint of deliverables and accountable for contributing to team productivity</td>
</tr>
<tr>
<td>TEST-DRIVEN DEVELOPMENT</td>
<td>A repetitive short software development process that, by the last iteration, results in the product meeting acceptable standards after passing a test defined by the associated user stories</td>
</tr>
<tr>
<td>THEME</td>
<td>A top-level objective that may span projects and products and be used at program and project levels to drive strategic alignment and communicate a clear direction; may be broken into sub-themes, which are more likely to be product-specific</td>
</tr>
<tr>
<td>THROUGHPUT</td>
<td>The amount of work effort produced within a given time</td>
</tr>
<tr>
<td>TIMEBOXING</td>
<td>A planning technique in which scheduled items are divided into a number of separate time periods (normally two to six weeks long), with each part having its own deliverables, deadline, and budget</td>
</tr>
<tr>
<td>USER STORY</td>
<td>A high-level definition of a requirement that has enough information for developers to approximately estimate the effort it takes to implement it; one or more sentences that captures what the user wants from the user’s perspective in plain language</td>
</tr>
<tr>
<td>VELOCITY</td>
<td>A relative number, derived from adding all story points of sprint, that represents a projection of how much work an Agile team can perform within a sprint, or the reporting of actual work being performed</td>
</tr>
<tr>
<td>VERTICAL SLICE</td>
<td>A breakdown of stepwise progressions for developing a user story incrementally</td>
</tr>
<tr>
<td>VISION</td>
<td>A strategic, high-level, series of simple statements that define a target group (users and customers), its needs (challenges the product will address), the product (three to five most important features) and value (benefits to the organization in developing the product)</td>
</tr>
<tr>
<td>VISION STATEMENT</td>
<td>A high-level product description explaining a product’s purpose, target customer, and any differentiators</td>
</tr>
<tr>
<td>WIP</td>
<td>Also known as work in progress, work started but yet to be completed</td>
</tr>
<tr>
<td>XP</td>
<td>Also known as Extreme Programming, this Agile method emphasizes frequent “releases” in short development cycles with adoption of new customer requirements that will improve software quality and responsiveness</td>
</tr>
</tbody>
</table>
APPENDIX B: FURTHER READING


Where it all started; the principles behind Agile by the group that founded the approach.


An excellent introduction to the principles and practices of Agile software development, this is a must-read when getting started with Agile and best of all, it’s free.


Excellent introduction to writing user stories.


Offers a detailed management framework that addresses everything from team building to project control, bridging the disconnect between the assumptions and techniques of traditional and Agile management, demonstrating why agility is better aligned with today’s project realities and how to simplify the transition.


Presents a roadmap detailing six core steps for migrating from plan-driven management to Agile project management and illustrates how they’re demonstrated in successful projects today.


Very brief but interesting approach to user story creation, focusing on format and content.


Describes user stories and demonstrates how they can be used to properly plan, manage, and test software development projects, highlighting successful and unsuccessful implementations of the concept, and providing sets of questions and exercises.


Good primer on estimation and how to be Agile with the planning and estimating activity. Cohn delivers practical information based on years of real-world experience bringing Agile into organizations. Provides lots of examples and case studies from software as well as non-software fields.


Discusses transitioning the organization from traditional software development methods to Agile.


Reminds Agile practitioners to respect all those involved in the project process.

Discusses why there does not have to be discord between the CMMI and Agile approaches to structuring software development maturity and identifies synergies that have the potential to dramatically improve business performance.


Excellent reference for Agile team leaders to keep retrospectives fresh and teams improving. A practical guide with tools and common sense tips for those practicing Agile. Useful techniques for each stage of the Agile retrospective.


Very brief but informative blog entry discussing the appropriate contents for an Agile project charter.


Discusses a controversial topic in Agile theory: What is the optimal length of time for a scrum?


Interviews, case studies, and summaries of all of the major Agile methodologies presented by one of the leaders of the Agile movement. Includes pointers on how to determine an organization’s Agile readiness, design a custom Agile methodology, and transform your business into a truly Agile organization.


Velocity, although a popular metric, is not without controversy; Highsmith, one of the founders of the Agile approach, takes issue with the measure in this article.


Offers a perspective on the culture of Agile projects suggesting “7 Cs.” (See Kumar for contrast.)


A framework for applying Agile principles and practices to release management and operations.


Presents a maturity model for building and releasing software; provides a structure for assessing your team or organizational capabilities and an approach for planning and executing improvements to existing practices.

Addresses the challenges of transitioning the federal government contracting process to Agile; brief, but offers some interesting, feasible solutions.


Offers a perspective on the culture of Agile projects suggesting “7 Cs.” (See Holler for contrast.)


An alternate and very good approach to calculating the velocity metric.


Provides a detailed examination and comparison of iterative methods including scrum, extreme programming, Evo, and the unified process. Offers a list of practical tips for adopting and running iterative processes, as well as an informative frequently asked questions section.


Detailed introduction to the user story process and results.


Like Highsmith (above), this author has some issues with the use of velocity as an Agile project metric, which he articulates in this article.


McHugh is a proponent of the velocity metric and describes versions of it in his three-part article; of those parts, two and three are the most important, however, all parts can be reached through this link.


A lighthearted look at applying the medieval bestiary concept to Agile project roles.


Offers an overview of the heated opinions of the role of product owner and scrum master and their combined (or separate) responsibilities.

Seeks to provide a set of metrics for Agile projects based on the tenets of Agile project management principles set forth in the Agile manifesto.


Discusses the application of EVM to Agile projects; refers to a number of alternate approaches, as well as a discussion of factors determining the applicability of the approach presented.


Provides what you need to know to get started with scrum. In 150 pages, the book covers the theoretical basis for scrum and how to organize teams, conduct daily scrum meetings, plan scrum projects, and track the backlog of items that need to be completed to finish a project.


Ideal book for an introduction to Agile methodology. A how-to guide that presents the basics of the scrum process and its artifacts from one of the initial signers of the Agile Manifesto, this book introduces the two key scrum concepts: iterative development and product backlog.


Written for the reader who understands scrum. Provides a framework for adopting scrum across the entire enterprise and some techniques for surmounting problems teams face when transitioning to scrum.


Well-written overview of the scrum approach to Agile.


Details the project management process areas and knowledge areas that must be addressed in transitioning to Agile.


Brief overview of the appropriate circumstances for employing Agile in software development.


Offers 10 tips for successful Agile software testing.


Discusses six principle elements to craft a good user story.

Step-by-step question-driven approach to developing user stories; very well written and useful.


A step-by-step guide offering a set of Agile meeting and facilitation checklists in an easy-to-understand and apply framework to help guide software development teams through the various Agile cycles.


A brief, but very good introduction to the use of the velocity metric in Agile/scrum.


Offers a bestiary parable explaining what Agile is and is not; this is the classic story often referenced by Agile texts to explain process role.
ABOUT THE AUTHORS

KEITH KERR
Keith has more than 20 years of experience helping organizations achieve a higher level of performance by applying program management rigor to their business processes. He has worked on dozens of programs in both private and government organizations. Currently, he is responsible for practice and solution development in project and program management and portfolio management. He holds an MBA in economics and public policy from George Washington University and is a certified scrum master, Project Management Professional® (PMP) and a graduate of the Defense Systems Management College program managers course.

JIM VINCENT
Jim is a senior management consultant with more than 25 years of experience in the arena of business and technical management, including project and program management, strategic and technical planning and execution, quality assurance, software verification and validation, computer systems analysis, and development of user manuals and technical documentation. He is the author of a dozen business and technical texts including Agile Project Management for the Accidental Manager (2 Vols.), Introducing Agile Project Management, A Guide to the Innovation Body of Knowledge, Auditing and Evaluating the Organization and Always Be Becoming: Change Management for the 21st Century. Jim has an MBA from Strayer University.