3

Acquisition & Management Concerns for Agile Use in Government Series

Management and Contracting Practices for Agile Programs

Software Engineering Institute
Carnegie Mellon University

Acquisition & Management Concerns for Agile Use in Government

This booklet is part of a series based on material originally published in a 2011 report titled *Agile Methods: Selected DoD Management and Acquisition Concerns* (CMU/SEI-2011-TN-002).

The material has been slightly updated and modified for stand-alone publication.

Booklet 1: Agile Development and DoD Acquisitions

Booklet 2: Agile Culture in the DoD

Booklet 3: Management and Contracting Practices for Agile Programs

Booklet 4: Agile Acquisition and Milestone Reviews

Booklet 5: Estimating in Agile Acquisition

Booklet 6: Adopting Agile in DoD IT Acquisitions

Management and Contracting Practices for Agile Programs

In this booklet, we discuss the characteristics and practices that are generally associated with contracting for and managing Agile projects. The viewpoint we take is primarily that of a project or program manager who is directly involved with oversight of an Agile development team.

Our research included instances of acquisition programs where a government employee filled this role, as well as examples where the person in this role was a contractor employee. We also differentiate, where appropriate, managerial behaviors of a manager on the acquisition side who is not directly supervising a development team.

The topics address the adoption of Agile using management and contracting practices best suited for use with Agile programs. Practices for continued execution are not specifically addressed in this booklet.

Common Agile Management Traits

The interviewees who provided source material for this booklet uniformly acknowledge that the Agile approach to project execution places demands upon all personnel that differ from other execution environments. The managerial role is uniquely affected by the features of the Agile approach.

The Agile program manager has more direct interaction with the development teams than is typical in a development project. This is in no small part due to the flatter organizations that comprise Agile development teams. In addition, the use of iterations demands that the program manager be more aware of and supportive of short-term planning. In large organizations, there may be more than one individual designated to fulfill the span of activities, in effect a management team. Note that in the role descriptions that follow, either multiple people could hold a role, or a single individual can hold one or more of the roles. This section takes inspiration from ideas advanced by Dean Leffingwell, then alters and expands those ideas to address inserting Agile practices into DoD acquisition [Leffingwell 2008].

Executing-Side Program Manager[1]

The managerial role for the organization executing a program exploiting Agile methods is distinguished by traits described below.

Leader. The program manager provides leadership and should spend more time with the team than behind the office desk. Consistent contact in this high-touch environment fosters team communication and cohesion. The “trust factor” is essential in the Agile environment, and the ability to delegate important tasks to others is essential. It is very helpful for the program manager to have a personality compatible with these kinds of practices.

[1] The executing-side manager could be a development contractor or part of an organic government team, such as an Air Logistics Center team.

Coach. The program manager acts as a coach, not a command-and-control supervisor. Instead of telling the team what to do, the coach seeds the team with ideas and allows the team to solve the problem on its own. For instance, the coach might ask, "Have you thought about how to do this better?"

This departure is driven by three features of this project environment:

- There is a need to help new people learn new behaviors, especially if they have not been exposed to it before.
- Established members of such teams will tend to be disciplined, able to act as mentors to new people, and require less traditional supervision.
- There is a need for collaboration, among team members and across teams.

Expeditor. The program manager must be vigilant in establishing an environment that fosters successful execution of individual iterations and the overall project. The discipline of the timeboxed iteration magnifies the effect of organizational and operational impediments. The more intimate involvement of the manager in the day-to-day program execution provides a basis for identifying operational impediments that reduce the probability of success, either through personal observation or by receipt of timely feedback from the development team.

Champion. The program manager must deal with upper-level management and stakeholders. For some time, part of the manager's job will be to provide adequate visibility into an execution model that will be unfamiliar, if not foreign, to upper-level management and other managerial stakeholders.

The program manager is likely to find it necessary to act as the translator to effectively communicate with this constituency. In at least one of the programs we interviewed, this was seen as one of the most critical elements of the managerial task. In fact, some have said performing the champion role will be the greatest challenge while adopting Agile development methods within DoD contractor organizations.

Ambassador. A key set of stakeholders are the prospective users and subject-matter experts whose sustained participation is necessary for successful execution. The program manager will need to cultivate relationships with these people and their leadership to ensure this participation.

These personas should be familiar to most program managers. The difference is the perspective from which the personas are implemented. Specific Agile training or certification would make it much easier to accomplish these roles in the Agile environment.

Acquiring-Side Program Manager

For the manager operating on the customer side, inside an acquisition organization, the above personas are still present, but with variations in emphasis.

Leader. The leadership role for the acquiring-side manager requires establishing and maintaining relationships with the leadership of the executing organization. The likely geographic distribution (not a particular attribute of Agile projects, but a fairly commonplace occurrence in today's programs) places additional demands on the acquiring organization.

As such, the acquiring-side manager may delegate representative(s) to be on site with the executing organization to maintain adequate visibility into the emerging product(s). Electronic means are also effective but lack the immediacy offered by physical presence.

In at least one of the programs we interviewed, the weekly presence of the acquiring-side manager was seen as a great benefit to the development team and the end users, because that individual could bring insight from the field and from the policy-making parts of the program organization to the development team in a timely fashion.

Coach. The acquiring-side manager serves as a coach for those people who are the direct contacts with the program in execution, such as the testing community, information assurance personnel, operations personnel, etc. That coaching spans the kinds of interactions the acquiring-side manager wants to have with the development team and the execution-side manager. It also includes defining the nature of the information and metrics the acquiring-side manager wishes to receive.

The acquiring-side manager has a direct stake in the nature and quality of the interaction between subject-matter experts and the development team, so the coaching role extends there as well. Much of this coaching will involve helping existing personnel make the transition to the fast-tempo, high-interaction environment that typifies Agile projects.

Expeditor. The acquiring-side manager has a significant challenge in efficient deployment of the people directly interacting with the development team and its management. For DoD projects, the norm for larger programs is that the development team will be geographically dispersed.

Identification of the best distribution of the available staff will be an ongoing challenge for the acquiring-side manager. Securing appropriate status information that does not unduly interfere with the tempo of Agile development will be a matter of negotiation and establishment of trust between the acquiring manager and the execution manager, as well as the staff doing the day-to-day work.

In addition, the norm is that end users and developers do not have much, if any, direct interaction. Finding and getting access for developers to the right end users in a timely fashion within a culture accustomed to separating these two groups is a challenge for DoD teams using Agile methods.

Champion. The acquiring-side manager is responsible for maintaining buy-in by external funders and stakeholders. It is unlikely that these stakeholders will be familiar with the dynamics of Agile projects, which places an additional burden upon the acquiring-side manager to provide a portrayal of project status and accomplishments that is accurate and to bridge the cultural gap.

In addition, the champion removes obstacles, rather than beating up the contractor for every risk that pops up or retracting award fees. As we said about the executing-side manager, some have said performing the champion role will be the greatest challenge while adopting Agile development methods within the DoD.

Ambassador. The acquiring-side manager must ensure the appointment of end users or subject matter experts to work with the developers. These could be proxies if actual end users are not available.

Product Owner

A distinct but important management role for the acquiring side is the product owner.

The product owner will

- define the features of the product
- decide on release date and content
- be responsible for the profitability of the product (return on investment [ROI])
- prioritize features according to market value
- adjust features and priority every 30 days, as needed
- accept or reject work results [Kovatch 2009]

Thus, one of the product owner's roles includes adjudicating conflicting requests for change. The product owner is responsible for the mechanism that ensures that conflicting user requested changes are resolved and that the system does not stray from its vision through a thousand little changes.

Lessons Learned Implementing Agile

Given the above management behaviors that are needed for successful management of Agile projects, there are some lessons learned from our interviews that particularly focus on management issues. We will discuss lessons in team building, increments/iterations in the larger scheme of the project, metrics, and geographic distribution.

Incentivizing Teams

A manager's role in team building, whether on the acquisition or execution side, is an important one and is reflected primarily in how the manager establishes trust with the development team and other stakeholders, and in the behaviors that the manager chooses to reward or punish. Honesty and open communication were key attributes for the manager that we heard over and over during interviews.

For example, in an Agile project, one of the goals is to understand the team's velocity with enough accuracy to plan iterations reliably. When this is achieved, not only does the team tend to deliver on time, they tend to do it without a lot of fuss. Rewarding the team effort that is exemplified by this case is appropriate as it reinforces one of the key tenets of Agile—collaborative teams. However, rewarding individuals who work abundant overtime to pull out a delivery, rather than addressing the root cause such as poor estimation of story points, will work against the team's ability to perform effectively.

Many team members said they liked being able to provide estimates that were realistic and see the success of their estimates at the end of the sprint. One reviewer stated that their first Agile implementation had a 50% increase in productivity. In addition, the manager's annual employee survey showed amazing improvements. This manager gave the team the authority to make decisions and took on the attributes that were mentioned under the executing-side manager section.

Another aspect of team building that is sometimes difficult for managers experienced in leading traditional projects is giving the team more autonomy in the selection of goals and in the selection and application of rewards. Agile teams told us that they found that the entire team understood and worked toward the selected goal. In addition, all team members understood and agreed to the expectations for each iteration.

Many managers are not accustomed to allowing the team to decide how it will distribute rewards or to self-organize into the roles that need to be performed to accomplish the project goals. Successful Agile teams that we interviewed all said that a focus on team awards such as team lunches and fun activities provided positive reinforcement. One team also said that 20% of the raise pool was given to the team to allocate.

One impediment to facilitating team building in a DoD environment is the common perception that team building exercises are a "waste of taxpayer dollars," not reimbursable and thus counterproductive. However, forming productive self-supporting teams requires some amount of focused team building. Collaborative, self-organizing teams are a key tenet for Agile. Therefore, a strong argument can be made for the use of team building exercises when forming or even continuing the use of an Agile team.

Mastering the Iteration

In Agile projects, the time box for accomplishing a useful unit of work usually refers to a 2–4 week period (iteration) where a small set of the functional requirements (expressed as user stories) will be moved through design into implementation and initial testing. These iterations have a different technical and business rhythm than most traditional projects are accustomed to accommodating and executing.

From a management viewpoint in particular, the movement of small elements of functionality/capability through the entire development life cycle in less than a month presents communication issues to middle/upper management accustomed to seeing more of a "complete the requirements, then complete the design, then implement" approach.

During the interviews one of the most often-cited benefits of Agile projects, for both end users and development teams, is the ability to accomplish something real in a short time, get feedback, and make course corrections soon enough to affect the overall outcome. Team members and end users get a sense of accomplishment, have an increase in team morale, and see actual usable software at the end of each iteration. The users found that they were getting what they asked for quickly and this encouraged them to continue using the product.

Some environments may have constraints such as rigorous configuration management, which can delay delivery to the customer. Even in these circumstances, demonstrations can be done to show the customers what they will be receiving.

DoD projects often have external requirements, like air worthiness certification or information assurance certification, that are not as easily accommodated within the iterations.[2] Thus, additional time is needed to meet the external requirements. Wherever possible, time between completion of a release's functionality and delivery to the end user is minimized.

The projects we interviewed have used various methods for addressing this. Sometimes they add an iteration at the end of a release cycle to deal explicitly with certification. Sometimes the certification activities are performed outside the development iterations framework altogether. Whenever possible, certification activities are included as tasks within an iteration. Especially when users or user surrogates are actively involved in the evolution of a capability, the delays that can occur due to external testing or certification, though necessary, may seem to be a hindrance.[3]

One interviewed program had a hybrid approach to certification. Some of the work was done before and after the iterations and some was done during the iterations. The sprint teams worked with the Information Assurance (IA) group before the iterations to define the latest security rules for their environment based on the latest guidance. Once the environment was defined, the teams would plan for their next iteration(s) within that environment.[4]

Their plans would be reviewed by the IA team for compliance before the iteration started and after the iteration finished. If at any time during the iteration the development team had questions about staying within the defined IA environment, the IA team would come in to assist. This process was very successful for them and they also passed an external audit of this process.

---

[2] There are myriad commands, services, DoD regulations, guidance, requirements, and reports that need to be considered depending on the project. Examples include DoD Architecture Framework, Standard Financial Information Structure, Federal Finance Management Improvement Act, service test and evaluation agency rules, and DoD Inspector General.

[3] The reader can perform an internet search on Microsoft Agile Security Development Life Cycle for an approach to certifications within an Agile life cycle. A detailed discussion of this topic is not included in this booklet.

[4] IA and other crosscutting special requirements define the environment in which a product is developed, integrated, and/or tested. They need to be dealt with ahead of the development effort.

From an acquisition manager's viewpoint, managing the rhythm of the iterations means negotiating windows of opportunity for planning actors such as user involvement, certification, and independent field evaluation so that those activities will contribute optimally to the tempo of the release. The acquisition manager also has a role in designing the technical milestones review schedule and its contents. The content of the technical milestone reviews have to be correlated to the content of the planned iterations. This particular area is sufficiently challenging that a subsequent booklet will be devoted to technical milestones in an Agile project.

Determine Appropriate Metrics

The old adage what usually gets measured, gets done is used to advantage in Agile projects. What gets measured in Agile projects, on the execution side, are:

- Elements in a product backlog (the master list of all functionality desired in the product usually grouped by user stories and epics—groups of related user stories) and the rate at which they are being addressed (typically called the "burndown rate"). While some of these items may never be implemented, this information could be used for trending data.
- The speed and effort related to completing one unit of work—usually a story point—an estimate of the complexity of individual stories that can be aggregated and used predictively. This is usually called "velocity" in an Agile project. However, there is a good bit of variation in exactly what is collected and how it is reported within the Agile project. This metric also varies per team, so care must be taken not to misuse or misconstrue the meaning of this metric. It is more appropriate for planning purposes.
- Progress toward release—a variation of the burn down rate that looks at completed stories versus estimated stories for a particular release to the field or end users
- Technical debt—loosely defined as the volume of lines of code that are poorly written, poorly refactored, do not follow coding standards, and are not supported with sufficient unit tests, and the amount of code duplication.[5]

Less technical debt is better. If technical debt is allowed to grow too big it could lead to unmaintainable code and eventually stop the entire development effort. The overall quality of the product is better with smaller technical debt.

Measuring technical debt helps achieve good quality and avoid undesirable results. Most Agile methods are supported by tooling that simplifies the collection of the raw data that goes into establishing the baselines for these measures. Some projects are small enough to use Excel spreadsheets, but most find that either the free tools widely available across the internet or commercially available licensed tools make the collection and use of appropriate measures efficient.

A note of caution is appropriate for metrics. Within the DoD environment, OSD

[5] http://software.intel.com/en-us/blogs/2009/05/29/agile-best-practice-3-of-10-measure-and-frequently-repaytechnical-debt/

and the individual services[6] use models to justify the life-cycle cost estimates. These models may use function points, RICE (reports, interfaces, conversions, enhancements or extensions) objects, or lines of code. Thus, the metrics would be different from those obtained from the Agile project using stories. Therefore, the program manager would end up keeping at least two sets of metrics by necessity. This is another disconnect (one not addressed further here) that needs to be resolved for the Agile project to run smoothly.

As with any measurements, using the data to punish individuals or teams is rarely productive. This is especially true in Agile environments, where this practice goes against Agile principles, where trust is a hallmark of successful projects and appropriate use of metrics to gain insight into the development process is crucial.

A special case for measurement that relates to estimation is measurement that supports a basis of estimate. In our booklet on cost estimation, we will discuss a frequently used measurement method in DoD projects, Earned Value Management, and its application in Agile projects.

Overcoming Challenges with Geographic Distribution of Projects

Some agilists insist that collocation is an absolute requirement for successful Agile projects; however, experience in both industry and the DoD programs we spoke with contradict this viewpoint if geographic distribution is managed effectively. In some ways, all the good management advice related to any distributed project applies here: ensure multiple communication pathways, have guidelines for when to move discussions among different communication modes, provide an appropriate amount of face-to-face time to promote cohesive team building, and so forth.

In addition, the information radiators,[7] which are unique to Agile implementations, provide a good source of communication. Of course, there will be times—such as for briefings to oversight groups—when the team or at least part of the team (PM, prime contractor, and user's representative) must be collocated.

Beyond the guidance that applies to any distributed project, there are some particular issues as described by some of our interviewees that could come up when using particular Agile methods:

- When using pair programming, try to avoid pairs that are geographically distributed. When they need to be distributed, supportive technology like webcams, telepresence software like Skype and GoToMeeting, and desktop sharing software, along with sufficient bandwidth to make it all useful, have been used in some of the programs we talked to.

---

[6] An OSD model is Cost Assessment and Program Evaluation [CAPE]. For the Army, the Deputy Assistant Secretary of the Army for Cost and Economics [DASA CE] uses the cost model.

[7] This term was coined around 2000 by Alistair Cockburn while standing in a Thoughtworks office, looking at all the paper on the walls around him. "Information radiator" refers to a publicly posted display that shows what is going on to people walking by. Information radiators work best when they are big, very easy to see (e.g., not online, generally), and change often enough to be worth revisiting. See http://alistair.cockburn.us/Information+radiator.

- Assigning a bounded set of functionality to one location whenever possible is a useful strategy. This division of labor follows the division of functionality, which takes advantage of the "natural" boundaries, and containment found when developing separate functionality.
- Synchronize iterations so that there is time for remote groups to access integration and regression testing environments. Often an extra day needs to be added to allow transfer, processing, and return of data (programs, actual user data, test scripts, etc.). Figuring out the right points for movement back and forth of data between teams is sometimes needed to ensure everyone stays productive.
- Continuous integration and regression testing environments are one of the biggest challenges for distributed teams. The physical act of moving large amounts of data to a central location for integration proved to be an initial issue for at least one interviewed project. However, in their expeditor role, interviewed managers who manage distributed Agile projects were key resources for teams in getting the right infrastructure and bandwidth for each site. Of course how a program intends to implement continuous integration and regression testing could also have security issues depending on the tools used and the security posture of the program.
- One program has a periodic face-to-face users' conference of all the teams and stakeholders involved in the project. The program characterizes the conference as "an essential element of their strategic integration of the using community into their development process." The agenda includes updating everyone on the current state of the product, its user base, release plans, customer testimonials, etc. It is a different type of interaction between the users and the developers, but one that is greatly appreciated by the entire team. Though expensive (rental of conference site, travel, time of participants), the program believes that its motivational value outweighs the cost by a good measure. The overall conference not only provided benefit to the developers in that they meet the users, but the user community also gets to meet the developers and asks questions first hand.
- One reviewer also suggested that sometimes you might choose to distribute teams deliberately to accomplish a specific goal. For example, when merging two companies in two different locations you might deliberately compose teams of people from both locations to start forming a single new company culture.

Establishing management behaviors and infrastructure that is tuned to Agile methods is a challenge, but the managers we interviewed found the effort to be worthwhile. Many of these managers took risks with their own managers to protect the Agile culture that they were building.The payoffs in terms of user satisfaction and productivity were unanimously considered successful. This "protecting the team and team culture" mentality is a critical part of being an Agile leader.

Contracting Issues and Solutions for Agile

One of our government reviewers said, "This is the single biggest barrier to the government operating like a commercial entity. We take 5 to 10 times longer to execute routine contract actions."

A particular management concern for Agile methods in the DoD, especially from the acquisition side, but also from the execution side, is the selection and implementation of appropriate contracting vehicles to support the types of practices that successful Agile projects exhibit. Due to the iterative nature of Agile and its propensity to accept (even welcome) change, many contracting vehicles present unique challenges for employing Agile methods.

A particular issue is the reporting and milestone requirements often levied against DoD contracts. These may be especially difficult to meet using Agile methods. For example, one of the interviewed programs was entering PDR. The biggest issue was to ensure the government review team understood what was included in PDR and what was not. Requirements assigned to iterations in the future would not have any design detail available, as opposed to a traditional PDR where everything would have some type of design available at PDR.

Another challenge is to construct any contract type so that it rewards working software and meaningful milestone review compliance rather than just traditional artifact production. Several of the interviewees stated that achieving rewards for working software that is balanced with sufficient documentation was a challenge.

This booklet is not meant to be an all-inclusive and exhaustive discussion of the contracting issues associated with employing Agile methods. Rather, it is only a brief introduction to the types of contracts and some associated issues. Four of the most common contracting vehicles are discussed in the following section, along with comments about what we found in our interviews in relation to the use of these contracting vehicles. One interviewee specifically stated that any contract type could be used; some were just easier to work with than others.

Cost Plus Incentive Fee (CPIF)[8]

CPIF contracts are used for a wide variety of purposes in the DoD. In terms of Agile development, a CPIF vehicle is a reasonable contracting vehicle to use when (as is often the case with Agile projects) requirements will be quickly evolving and redirection of the tactical details of the software is expected. The incentive fees can be constructed around measures and motivators that support Agile methods, if the acquisition manager is cognizant of the kinds of management and measurement aspects, as we have described above.

The management team of one of our interviewees credits its CPIF contract structure as a key success factor. The program delivers multiple releases per year of ever-improving functional requirements despite having zero software requirements specified in the enabling contracts.

This is possible because the contract is a services type contract for software engineering support. This allows the development team to avoid situations where

[8] For information on contracting vehicles discussed here, as well as other contract types typically used in the DoD, please refer to http://www.dtc.dla.mil/dsbusiness/Info/contracts1.htm or http://guidebook.dcma.mil/18/ContRecRevconttypes.htm.

a contractor is obligated to satisfy a requirement that has become irrelevant or obsolete. The team can respond to changing needs and priorities without onerous contracting actions.

Fixed Price (FP)

There are a variety of fixed price contracts: firm fixed price, fixed price incentive, fixed price with economic price adjustment, fixed price redeterminable, and fixed price level of effort.[9] The government prefers this type of contract because it encourages the contractor to contain costs. Fixed price contracts are used with Agile projects in industry, but usually after the product backlog and critical elements for each release are defined. In industry, the contracts are generally short in time span in comparison to what is typical in the DoD.

Indefinite Delivery Indefinite Quantity(IDIQ))/Delivery Orders

IDIQ/Delivery orders, or task orders, comprise the largest percentage of contract type in terms of the programs that we interviewed. Some of the benefits were that IDIQ vehicles provided the most flexibility for adjusting to changing operational needs and provided more opportunities for close working relationships between the developers and the end-user community they are serving.

Although none of our interviewed programs expressed it exactly this way, IDIQ types of vehicles can be managed more like a service contract, where the service being provided is on-demand software evolution, rather than like a product contract. Some of the author team sees this framing of software development as a productive service for Agile projects in the DoD, one that is supported by the prevalent use of the IDIQ contract type.

However, just because IDIQ provides flexibility, the government still needs to require an estimate in cases where a release is mapped to (or the subject of) a task order. The releases closer to being performed should have better estimates than those further out in the schedule. That is, the immediate releases will have deliberate estimates and those further out will have planning estimates. If the estimates change, the contractor needs to provide a reason. Otherwise, the contractor may not look ahead further than a single increment.

One potential frailty of the IDIQ contracting model, particularly for a large overall program, is the management of multiple delivery order contracts running in parallel. Even if the art of the iteration has been mastered by the team(s) involved, there is a potential for the focus of program management to be limited to each of the delivery order contracts in isolation, particularly if metric reporting is limited to the scope of individual delivery orders. Make sure that oversight is across the entire program rather than focused on individual tasks, and keep a continuing focus on quality and integrity within product management rather than just looking at the project management detail. While this may seem obvious to the seasoned program manager, remember that your team is learning how to work with Agile and may need to be reminded of the basics.

[9] http://guidebook.dcma.mil/18/ContRecRevconttypes.htm

Time and Material (T&M)

T&M contracts are used when it is not possible to estimate accurately the extent or duration of the work or to anticipate costs with any reasonable degree of confidence.[10]

From a government perspective, this type of contract requires significant oversight to assure that the contractor is performing efficiently and using effective cost control measures. From a contractor perspective, it fits rather well with the typical Agile planning and estimation process. T&M contracts may be the most successful of the contracting types discussed here for use with Agile. In fact, many of the interviewees, both contractor and government, commented that T&M contracts were the easiest to use.

Contracting Vehicles Summary

Different contracting vehicles can work for Agile methods. Acquirers must be savvy about the interpretation limits of different contracting regulations that affect the type of contract they are contemplating. In cases where we saw awareness of differences in the cultural norms of Agile and traditional acquisition teams, the contract vehicle chosen was not an impediment to getting the desired results using Agile. In other words, the contracting personnel were aware of how Agile methods worked: change is part of the norm, not an exception, they knew what would be delivered when using those methods, and they adjusted their expectations accordingly within the limits of the contracting rules and regulations.

Conclusion

Contracting for and managing Agile projects within the DoD environment requires adopting particular practices in order to realize the full benefits of Agile methods. While Agile places demands upon all personnel that differ from other execution environments, the practices themselves are entirely consistent with the current regulatory and policy environment.

Project and program managers responsible for overseeing Agile teams can therefore chose from a variety of standard contract types, depending on their requirements, timeline and other factors. This booklet aimed to provide some initial insight into when and why to use each one. Similarly, this booklet set out to introduce the various roles program managers may be called on to fulfill (leader, coach, champion, etc.) from the executing-side and acquiring-side alike.

[10] http://guidebook.dcma.mil/18/ContRecRevconttypes.htm#Time and Material

References

[Kovatch 2009]
Kovatch, D. Roles & Responsibility of the Product Owner. Scrum Gathering, Orlando, FL, 2009. Scrum Alliance, 2009. http://www.scrumalliance.org/resources/617

[Leffingwell 2008]
Leffingwell, D. Scaling Software Agility. Addison-Wesley, 2008.

This material is based upon work funded and supported by the Department of Defense under Contract No. FA8702-15-D-0002 with Carnegie Mellon University for the operation of the Software Engineering Institute, a federally funded research and development center.

DM17-0009

About the SEI

For more than three decades, the Software Engineering Institute (SEI) has been helping government and industry organizations acquire, develop, operate, and sustain software systems that are innovative, affordable, enduring, and trustworthy. We serve the nation as a federally funded research and development center (FFRDC) sponsored by the U.S. Department of Defense (DoD) and based at Carnegie Mellon University, a global research university annually rated among the best for its programs in computer science and engineering.

Contact Us

Software Engineering Institute
4500 Fifth Avenue, Pittsburgh, PA 15213-2612

Phone: 412.268.5800 | 888.201.4479
Web: www.sei.cmu.edu | www.cert.org
Email: info@sei.cmu.edu