ASI GOVERNMENT

VIRTUAL ACQUISITION OFFICE™ ADVISORY

March 2014

Enabling Acquisition Success for Agile Development

By Frank McNally

This Advisory *introduces the agile development philosophy and life cycle and explores acquisition strategies that enable its success.*

To federal leaders concerned that their next information technology (IT) project will suffer the fate of recent and well-documented technology failures, the concept of agile development holds great promise. Recognizing the philosophy's potential to significantly improve the outcomes of IT projects and bring them in on schedule and within cost, the Office of Management and Budget (OMB) has urged agencies to favor agile development over the "grand design" approach of traditional waterfall development.[1]

But what exactly is agile development? How does it differ from traditional software development methodologies, and how can you enable its success through the acquisition process? This *Advisory* begins to explore these and other questions related to the implementation of agile development within the federal government. We start with an introduction to the agile philosophy and the elements of the agile development life cycle before looking at acquisition strategies that enable its success. This foundation will increase your understanding of agile development and position you to support its implementation at your organization.

What is agile development and how is it different from traditional methods?

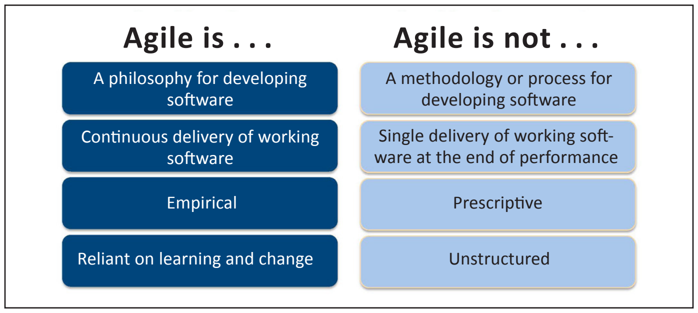
Agile refers to a philosophy for developing software through iterative "bursts" of effort, where all members of an acquisition team work together to deliver functionality throughout the project as opposed to all at once (and usually at the end). By delivering working software incrementally, agile takes an empirical approach by using lessons learned and input from users and stakeholders to determine the pace and performance of the project team. Agile is not, as shown in the chart on page 2, based on prescriptive specifications that attempt to predict the requirements of an IT product or service in total and before starting work. However, while agile has the capacity to change plans and incorporate learning, it is not completely unstructured.

The traditional waterfall methodology of software development is a sequential and linear process where requirements gathered in a preaward environment strictly govern contractor performance in the postaward environment. It is characterized by a long planning process resulting in highly detailed specifications and stringent work plans that dictate all facets of the development cycle. Once completed, the product is delivered to end users in one "big bang" deployment. Where agile uses an evidence-based or empirical approach, waterfall requires significant front-end planning, detailed (and often inflexible) requirements documentation, strict project planning, and limited input from end users or customers. And where agile delivers working software continuously throughout the development life cycle, waterfall provides software at the end of the development period in an all-or-none delivery.

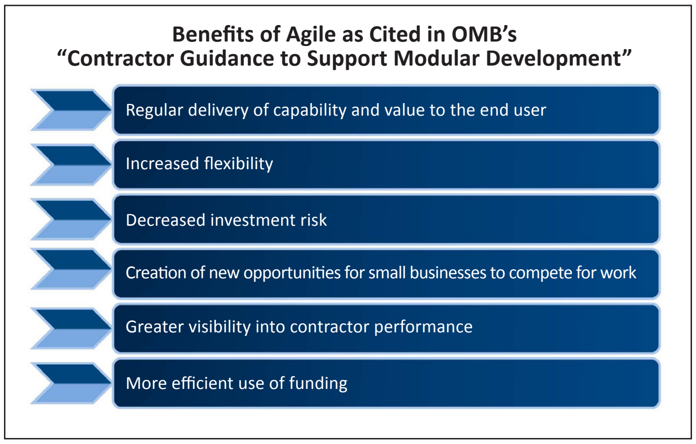
What are the benefits of using the agile development philosophy?

The benefits of agile development have the potential to far exceed those of traditional waterfall while providing a more manageable risk environment. Today, fewer resources are required to develop and test new software, and data analysis makes it possible to determine in near-real time whether it works. Errors can be fixed in hours with relatively little interruption to end users, allowing organizations to constantly upgrade their IT environment by adding new features to existing software on a perpetual basis.

Given these trends, OMB has encouraged agencies to adopt agile principles by breaking large requirements into smaller modules, to deliver functional, useful software throughout the period of performance. This makes good business sense given Moore's Law, which observes that computing power doubles approximately every 18 months.[2] The benefits of agile development (outlined in the chart at right) are consistent with the principles of modular contracting as described in Federal Acquisition Regulation (FAR) part 39 and are a focal point of this *Advisory*.



Agile is . . .	Agile is not . . .
A philosophy for developing software	A methodology or process for developing software
Continuous delivery of working software	Single delivery of working software at the end of performance
Empirical	Prescriptive
Reliant on learning and change	Unstructured



Benefits of Agile as Cited in OMB's "Contractor Guidance to Support Modular Development"

- Regular delivery of capability and value to the end user
- Increased flexibility
- Decreased investment risk
- Creation of new opportunities for small businesses to compete for work
- Greater visibility into contractor performance
- More efficient use of funding

As an acquisition professional, what do I need to understand about agile?

First, understand that agile is neither a methodology nor a specific set of processes. While it is informed by principles and guidelines, agile development is more of a philosophy, and its implementation is dependent on the experience and talent level of its practitioners. Therefore, acquisition professionals need not be expert in all things agile. Simply understanding a few fundamental aspects will enable you to support an agile procurement while providing a foundation from which you can build mastery of the technical nuances of this discipline through continued experience.

Second, it is important to understand the key values of this philosophy, as described in "The Agile Manifesto."[3] Agile development values:

- **Individuals and interactions** over processes and tools
- **Working software** over comprehensive documentation
- **Customer collaboration** over contract negotiation
- **Responding to change** over following a plan

While agile values the attributes on the left over those on the right, it does not advocate a complete abandonment of tools, documentation, negotiation, and planning. Rather, the values of agile development are informative for the philosophy and its application to the dynamic nature of today's IT environment.

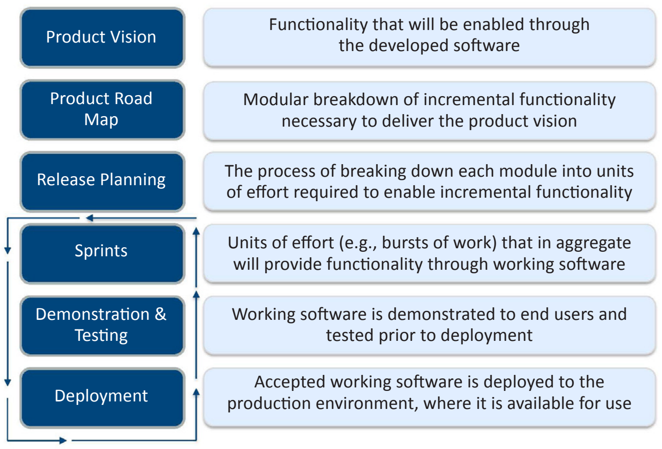
What do these values mean for the acquisition professional?

Valuing individuals and interactions over processes and tools is part of agile's empirical focus. Through a modular approach to software development, users help design, develop, and test working software throughout the period of performance to confirm that functionality has been delivered. Business acceptance (of the functionality being delivered) is critical to success in agile, so subject matter experts and senior program managers alike should expect to work closely with the development team on a daily basis. The processes used to deliver functionality are not important and cannot be predicted, so detailed specifications and product schemas are replaced with user input, acceptance criteria, and, ultimately, user acceptance. This is a dramatic shift from the robust requirements and lengthy specifications characteristic of waterfall development and will ease the burden of the acquisition team by reducing the paperwork and planning required to develop working software.

By collaborating with customers, agile developers gain an understanding of end user needs and use that understanding to deliver software that supports functionality. Creating detailed specifications and project plans can be wasteful and can prevent practitioners from incorporating feedback and lessons learned early in the life cycle in future development phases. When the developers learn an easier way to design software features, they may change their habits to leverage this efficiency because agile values responding to change.

These values require a different mind-set for acquiring IT products in the federal acquisition environment. Now that we have an understanding of the agile philosophy, let's examine the basic concepts of the agile development life cycle. The sidebar at right defines the terms common to the agile life cycle, so you can familiarize yourself with this terminology before reading further.

Agile Terminology



Product Vision	Functionality that will be enabled through the developed software
Product Road Map	Modular breakdown of incremental functionality necessary to deliver the product vision
Release Planning	The process of breaking down each module into units of effort required to enable incremental functionality
Sprints	Units of effort (e.g., bursts of work) that in aggregate will provide functionality through working software
Demonstration & Testing	Working software is demonstrated to end users and tested prior to deployment
Deployment	Accepted working software is deployed to the production environment, where it is available for use

What is the agile development life cycle?

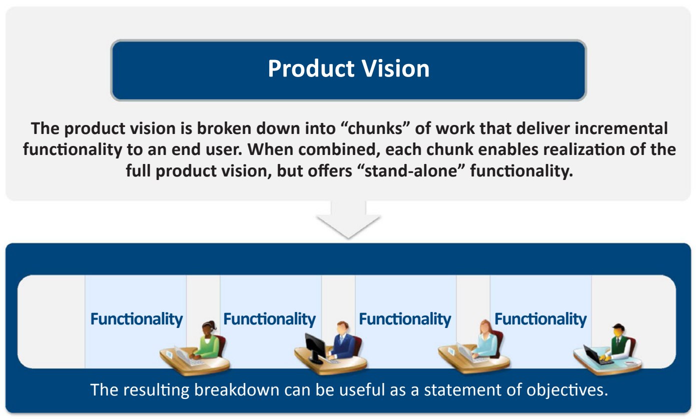
The agile development life cycle starts with a product vision that describes at a high level the functionality of the software that will be developed. The product vision is useful as a statement of need and serves as a guide for breaking work into functional elements. When a product vision is broken down into functional elements, it provides a statement of objectives (SOO) or statement of work (SOW) that allows the agile development team to further define the level of effort and tasks required to deliver working software in support of that functionality. (See the graphic below.)

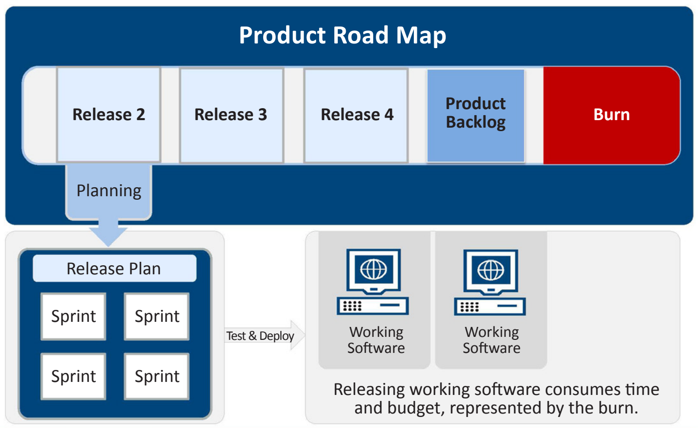
After the product vision is broken down into functional elements, the agile team conducts preliminary planning to determine how many releases of working software will be required to deliver the functionality described in the product vision. This effort results in the product road map, which demonstrates how the efforts of the development team will yield continuous delivery of working software to the end users. Keep in mind the product road map is a living document; it is intended to evolve throughout the development effort as feedback and lessons learned are gathered by the agile team.

How does the product road map support agile implementation?

The product road map is used by the agile team to conduct "just enough" planning to determine the specific effort, coding, and engineering tasks that will be required to produce working software.[4] Each functional element envisioned in the product road map is converted to a release of working software, but only one release is planned at a time, to allow the development team to incorporate lessons learned into future releases.

The effort required to provide functionality through the delivery of working software is called a "sprint," and a release plan can include as many sprints as are required to deliver working software, within the schedule and budget resources allotted to the release. Therefore, a release plan consumes budget and schedule resources available to the project; as additional releases are planned, they continue to consume schedule and budget resources in a process known as "the burn." The graphic below shows how each release of working software consumes schedule and budget, in effect "burning down" the product road map.


Product Vision
The product vision is broken down into "chunks" of work that deliver incremental functionality to an end user. When combined, each chunk enables realization of the full product vision, but offers "stand-alone" functionality.
Functionality
Functionality
Functionality
Functionality
The resulting breakdown can be useful as a statement of objectives.



As the agile team completes each release, it is tested against a set of acceptance criteria and metrics established during the release planning process before being deployed in the production (e.g., live) environment where it is available to end users as working software. Additional features that support the product vision or bug fixes for released software are maintained in the product backlog, where they can be incorporated into releases that finish ahead of schedule or into an entirely new release (presuming that schedule and budget resources are available, as is the case with "nice to have" features).

The agile team continues to plan each release individually, complete the work through sprints, and deploy working software to the production environment until the functionality has been achieved or all schedule and budget resources have been consumed. To gain a better understanding of the consumption of resources in agile development, let's consider what it is we are actually buying in an agile acquisition.

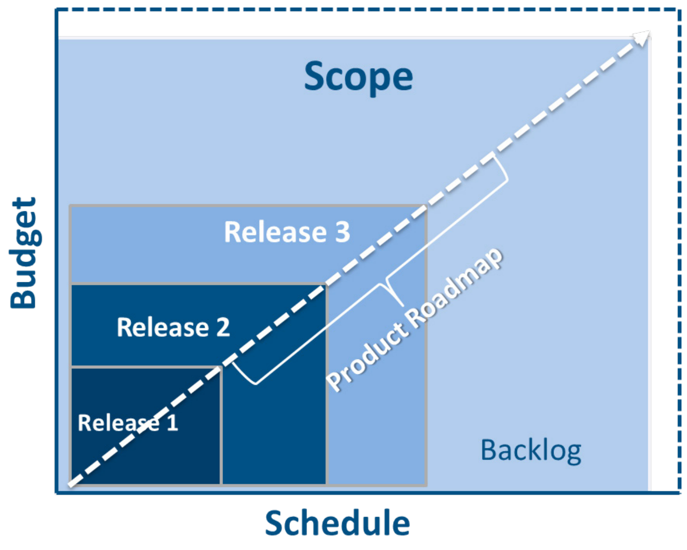
What are we buying in agile development?

In agile development, we are buying effort (e.g., services) in support of functionality, which is continuously delivered through the release of working software. Think of the scope of an agile acquisition as the creation of functionality in support of the product vision within the constraints of schedule and budget. Any feature or software that can be created and delivered within the constraints of schedule and budget should be considered within scope. The graphic at right depicts the concept of scope as functionality.

Notice in the graphic how each release consumes available budget and schedule resources while delivering functionality along the product road map. With each release, the product backlog shrinks, as do available budget and schedule resources. As long as the additional features support the functionality described in the SOO and are delivered within schedule and budget constraints, they may be considered within the scope of the contract. This alleviates a primary obstacle to enabling agile success through current acquisition methods, although it will require a degree of cultural change within the organization to accept this principle of "scope as functionality."

One final note about what is being procured through agile development: You are buying services in support of a product, not the product itself, because the product can change as the organization's needs change. New features can be added or obviated based on lessons learned by the agile team, so the exact features and functionality of the end-state product should not be fully defined. These features (and the end-state product itself) are subject to the learning that occurs during each release of working software and the efficiency of the development effort. As we have described, agile development has the ability to deliver functionality in excess of the original vision, so long as that functionality supports the product vision. This will have an impact on how you choose to structure your acquisition strategy, as we detail in the following sections.

Scope as Functionality



Acquisition Strategies to Enable Agile Development

Now that we have reviewed the agile development life cycle, we are ready to consider acquisition strategies that enable agile development. The best opportunity to promote agile success is to design an acquisition strategy that maximizes the realization of the product vision within the constraints of schedule and budget. To develop an appropriate acquisition strategy, we recommend using the practices described in FAR part 39, "Acquisition of Information Technology," to the maximum extent practicable.

A primary component of part 39 is the use of modular contracting strategies that address the risk of technical obsolescence inherent to the IT environment. You no doubt have experienced technical obsolescence in your personal life; the flat screen television or smartphone you purchased last year lags the technical features of today's model. In acquiring IT for the federal government, obsolescence is a primary concern and discourages the use of waterfall development, where all features and functions are deployed at the end of the product development cycle. Modular contracting is beneficial to acquiring IT, and your strategy should seek to enable these benefits by employing the principles described in FAR part 39.

What considerations are useful in creating a successful acquisition strategy for agile development?

In its June 2012 report, "Contracting Guidance to Support Modular Development," OMB recommends three strategies for enabling agile development: indefinite-delivery vehicles (IDVs), stand-alone contracts, and successive contracts. Each strategy has its own utility, and the selection should be informed by the characteristics of your software development needs. For an enterprise-wide software deployment effort, an IDV offers the advantage of consistency among the program team and the ability to divide the product vision into smaller chunks (e.g., modules) that can be delivered through task orders. In addition, an IDV can enable the use of multiple contractors for different elements of the product vision, or provide for a systems integrator that can provide project management assistance to the overall program.

A stand-alone contract is useful for smaller products that can be developed in the near term and do not require extensive integration with legacy systems. A stand-alone contract will require more support from the acquisition team to support the agile development effort and make end users available to inform the release planning process. For products that are more complex in nature, or will require extensive integration with legacy systems, a successive contract strategy that adheres to the guidelines of FAR 39.103, "Modular Contracting," may be required. If successive contracts are envisioned, you should consider the procurement life cycle at your agency to determine how quickly those successive contracts can be awarded. If too much time elapses between contracts, the progress of the agile development effort will suffer.

What considerations will inform the selection of contract type?

Once the acquisition strategy is completed, you will need to determine the type of contract that will be used. Unfortunately, there is no formula for selecting the right contract type, as it depends largely on the nature of your requirement and the placement of that effort within a broader program or agency requirement. Our thoughts on several of these options follow:

- **Fixed-price contracts** can be challenging because they elicit more of the "big bang" procurement approach, where vendors are asked to deliver a predetermined contractual outcome (e.g., software product) that often is influenced by rigorous price competition. Fixed-price contracts may have an adverse impact on the agile development life cycle by inhibiting the incorporation of lessons learned into subsequent release planning, because unforeseen challenges must be accommodated through contractual administration (e.g., modifications and change orders). This can lengthen the time between releases and reduce the return on your software investment given the risk of technical obsolescence.
- **Cost-type contracts** offer the advantages of schedule and cost flexibility, which are important in an agile development effort. Cost-type contracts also can be incentivized to motivate performance, and can enable functionality to be delivered at a lower cost than a fixed-price contract because the government has more control over funding decisions and the progress of development. Cost contracts are well suited for agile development because they allow the incremental delivery of working software, though they require a level of experience and expertise from the acquisition team and strong support from acquisition leadership. Cost-type contracts also have their own unique requirements for cost analysis [as described in FAR 15.404-1(c)], though these requirements offer significant advantages over an evaluation based on total product price. Also keep in mind that cost-type contracts require additional administration and oversight because contractors have less incentive to control total costs. Applying an award or incentive fee structure to contractual performance will mitigate this risk, but such structures do require significant postaward surveillance.
- **Time-and-Materials and Labor Hour contracts (TM/LH)** are worth consideration, given the focus of agile development on the continuous delivery of functionality. TM/LH contracts allow the government to purchase a not-to-exceed (NTE) amount of service in support of the overall product vision. This is consistent with our guidance that the scope of an agile development effort should be bound only by schedule and budget constraints. As long as the NTE amount is not breached, a TM/LH type contract for agile development can enable the delivery of functionality

FAR Guidance at 39.103(e)

To avoid obsolescence, a modular contract for information technology should, to the maximum extent practicable, be awarded within 180 days after the date on which the solicitation is issued. . . . To the maximum extent practicable, deliveries under the contract should be scheduled to occur within 18 months after issuance of the solicitation.

in excess of what was originally estimated, so long as it can be linked to the product vision. Keep in mind that TM/LH contract types require the acquisition professional to document in a determination and findings document why no other contract type is suitable. Therefore, a thorough review and analysis of FAR subpart 16.6 should be conducted prior to selecting this contract type. In supporting your determination and findings document, consider using an incentive structure whereby future business (perhaps in the form of sprints or additional releases) can be earned through effective management and reporting of hours burned. This offers a way to control costs and promote efficiency, thereby addressing a frequent criticism associated with the use of this contract type.

What are some other considerations when determining acquisition strategy and contract type?

Incentive fees and other structures such as award terms have utility for agile development. These incentives can be predicated on successful completion of individual releases, which can be included in modular contracting structures described in FAR 39.103. Agile development does require an element of consistency throughout the process, so incentivizing performance through the promise of future work not only supports government surveillance but also motivates the contractor to work within time, schedule, and technical constraints.

For larger efforts, an IDV approach provides flexibility to award task orders for each release plan. If your total value is under $108 million, you can defend a single award indefinite-delivery, indefinite-quantity (IDIQ) contract, but beyond that threshold, a multiple award IDIQ is likely to be required.[5] Plan accordingly, as all awardees must have fair opportunity for consideration of task and delivery order awards throughout the contractual life cycle, and this will increase your procurement cycle time. This can affect the execution of future task orders, which is important because consistency and efficiency are essential to realizing the benefits of agile development.

Hybrid contract types hold promise for agile software development. IDVs enable hybrid contract types, where fixed price orders can be let for project and release planning exercises. Then, TM/LH or cost-reimbursable contract types can be used for the actual development of working software. If working with firm fixed price type contracts, you may explore the possibility of including language within the contract that enables the conversion of contract type to something more suitable for supporting release planning and execution, such as cost or TM/LH. Conversely, cost or TM/LH type task orders can be let for early releases until a better understanding of schedule and budget requirements is gained, at which point fixed price orders can be let for subsequent releases.

A Note on Section 508

When developing IT products and services, it is necessary to comply with section 508 of the Rehabilitation Act of 1973, which addresses accessibility for individuals with disabilities. The integrated product team member representing your Office of the Chief Information Officer should be sure to coordinate with your agency's section 508 compliance office when developing requirements for IT products and services.

For smaller efforts, single award contracts can address the procurement cycle time issue, but then the requirement must be developed to enable the progression of release plan execution. In these structures, priced options, incentives, and award terms can be useful tools to monitor and motivate contractor performance. If awarding to a small business, you also can take advantage of OMB's guidance to accelerate payments (within 15 days of performance targets) to the small business so that it has the working capital to continue its development efforts.[6]

The bottom line consideration when selecting contract type is that it should be driven by the goal of enabling ordering efficiency, or efficiency in continuously delivering working software. Ideally, you can award contracts with periods of performance that are 6, 12, or 18 months in duration (consistent with FAR 39.103), where the end deliverable is working software described by the product vision. Consider the culture and history of your agency when selecting your contract type. Whatever your choice, you must ensure the rapid and efficient execution of contracts or task orders for the development of working software while managing performance so that outcomes, rather than process, are the determinants of successful delivery.

Are performance-based acquisition techniques appropriate for agile?

Absolutely. Agile development projects require a scope of work that allows for flexibility and change, so you can add great value throughout the acquisition process by using performance-based acquisition (PBA) techniques. In addition, because agile development uses "just enough" planning to deliver working software in support of the product vision, PBA allows the acquisition professional to focus on functionality rather than complex requirements definition and strict adherence to project plans.

In the planning phases, the acquisition professional should require an integrated project team (IPT) inclusive of program owners, end users, and IT professionals who will work together to create the high-level product vision and ensure consistency with your organization's IT environment. In supporting the IPT during the planning phase, the acquisition professional should focus on defining the constraints of schedule and budget so the IPT understands the resources available to the project. This informs the identification of features that must be delivered to achieve the desired functionality within those constraints. Additional features can be delivered when allowed by schedule and budget, and avoiding the prescriptive requirements characteristic of waterfall development provides the agile team with more flexibility to deliver functionality in the most efficient manner.

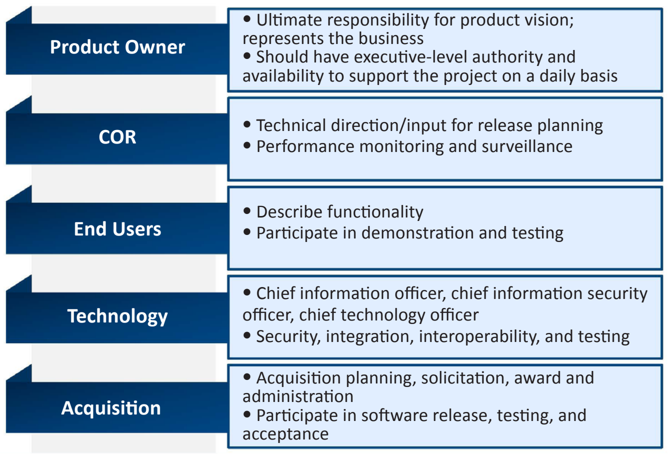
In addition, when using PBA to acquire IT services under FAR part 39, you are directed to neither prescribe nor require minimum experience or educational requirements for proposed contractor personnel.[7] This is important because we rely on the expertise of our agile vendors to propose the latest and greatest skills necessary to deliver working software. They should have the autonomy to provide resources capable of enabling the product vision based on their experience in software development, not ours.

Who should be included in the IPT?

Agile software development requires constant and continuous collaboration, not just among developers and project owners, but with all stakeholders of the project. OMB recommends assembling an IPT with well-defined roles and responsibilities. Given that a fundamental requirement for agile success is reliance on collaboration, we recommend taking advantage of the acquisition planning process to ensure all appropriate team members are identified, and have them (and their supervisors) commit time and resources to supporting the project by contributing to and signing the IPT charter.

While the composition of your IPT will depend on your specific circumstances, certain roles must be included. The product owner is the most important, because this individual represents the business. The product owner should have executive-level authority and available bandwidth to support the project on a daily basis, because he or she will be required to make decisions that affect the ultimate functionality of the working software. Additional roles that must be included in the IPT are outlined in the graphic above. It includes a representative(s) from the Office of the Chief Information Officer (OCIO) who will be integral in thinking through the technical and infrastructure issues involved in building, testing, and delivering software in smaller increments.

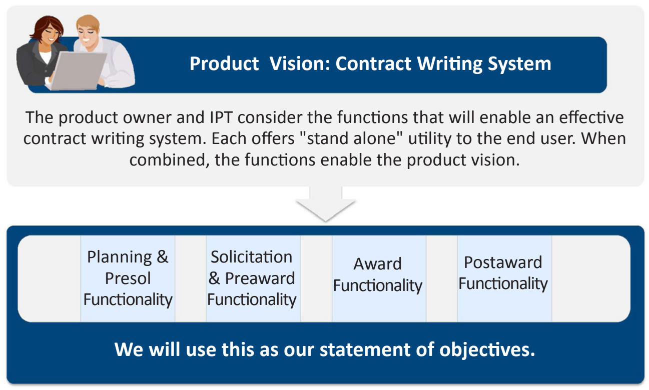
The Integrated Product Team



Product Owner	• Ultimate responsibility for product vision; represents the business • Should have executive-level authority and availability to support the project on a daily basis
COR	• Technical direction/input for release planning • Performance monitoring and surveillance
End Users	• Describe functionality • Participate in demonstration and testing
Technology	• Chief information officer, chief information security officer, chief technology officer • Security, integration, interoperability, and testing
Acquisition	• Acquisition planning, solicitation, award and administration • Participate in software release, testing, and acceptance

How does the product vision support development of a statement of objectives?

As illustration, let's consider how we would use PBA to procure a contract writing system with an agile development philosophy. The agile development life cycle starts with a high-level product vision that describes the functions that will be delivered through working software. A good practice in developing the product vision, which is driven by the product owner, is to start by describing at a high level the functionality required. Once a high-level functionality is written, the IPT can break that down into a product vision that describes the functions required to deliver ultimate functionality, as depicted in the graphic below. In this case, we created a SOO that decomposes the high-level functionality described in the product vision into four functional elements (planning & presolicitation, solicitation & preaward, award, and postaward) that must be developed to produce our contract writing system.


Product Vision: Contract Writing System
The product owner and IPT consider the functions that will enable an effective contract writing system. Each offers "stand alone" utility to the end user. When combined, the functions enable the product vision.
Planning & Presol Functionality
Solicitation & Preaward Functionality
Award Functionality
Postaward Functionality
We will use this as our statement of objectives.

How do I use the SOO?

Rather than prescribe to a developer how it should create the software required to enable functionality, PBA encourages vendors to demonstrate the results of their efforts. In this manner, PBA supports the principle of "just enough" planning by describing desired functionality in a SOO and then having vendors propose a performance work statement (PWS) against that SOO. This preserves the ability for the agile vendor to leverage its experience while providing the opportunity to incorporate lessons learned into the production cycle. Strictly defined requirements documents do not enable such flexibility and thus are barriers to the full realization of agile benefits.

While the SOO should focus on outcomes, it also should include constraints and environmental conditions relevant to your IT and program environment. It is within these constraints that the agile development team must operate, so you must describe any legacy software development requirements, security policies, and relevant IT policies or procedures, as they will have an impact on the development of working software. Your IPT will be especially helpful in identifying these constraints, and is a primary reason why representatives from the OCIO are invaluable members of the agile IPT.

Does the SOO bind the agile team to deliver specific requirements?

No, the SOO should not bind the agile team to deliver a specific set of functional requirements. This is important because if the desired functionality is delivered ahead of schedule and within the allotted budget, the agile team can use the remaining time allotted for that release to develop additional features in support of the product vision. As the agile development team works to deliver functionality in support of the product vision described in the SOO, the acquisition professional should focus on two primary constraints: schedule and budget. In this manner, an agile development effort has the potential to deliver functionality in excess of what initially was envisioned by the product vision within the contractual limitations of budget and schedule, rather than only the prescribed functionality predicted in a traditional waterfall development requirements suite.

A SOO that breaks the product vision into functional elements will enable vendors to propose a PWS that demonstrates the result of their development efforts through the continuous delivery of working software. Upon award, the PWS is used to guide the development effort through creation of the product road map, which guides the release planning activities that deliver functional outcomes described in the SOO. Agile development's goal of delivering maximum functionality in support of the product vision within the constraints of schedule and budget essentially requires that we include in the solicitation the budget available to the project, which can be expressed as a ceiling amount or NTE amount. As we discuss in the next section, this presents a unique challenge to the evaluation of vendor proposals.

Evaluation of Proposals

A primary challenge that must be addressed in acquiring agile development is the notion that vendor-proposed PWSs do not enable an "apples to apples" comparison of proposals. Each vendor-proposed PWS will be unique; some will propose more releases than others, but so long as the functionality being delivered can be achieved, this is not a concern. However, it does put pressure on the evaluation process and necessitates a creative approach to technical and price evaluation.

Let's recall the contract writing system example as we consider an approach to evaluating vendor-proposed PWSs. A good place to start thinking through this is to identify the primary categories of proposal evaluation: technical, past performance, and price.

Technical Evaluation for Agile

For our contract writing system, we want to differentiate between vendor proposals in terms of their technical understanding and capability to continuously deliver working software. A good place to start is to consider how their PWSs support the product road map, and whether the order of functionality-to-be-delivered makes sense for the end-state working software. For instance, if a vendor proposes to deliver working software in support of contract closeout before delivering award functionality, this may indicate a lack of technical understanding and increased technical risk (unless accompanied by supporting logic).

In addition, we should request that vendors include in their proposals a description of how they manage the implementation of their own agile principles. What is their technique for release planning? Do they have a plan to engage with end users to understand the features that must be delivered in support of the functionality envisioned by that release? How do they capture lessons learned relevant to the software development effort and use those lessons to increase the efficiency of future releases? Key elements to consider are their use of collaborative forums for sharing information across the team, their testing and quality assurance process, their approach to configuration management, and their appreciation for engaging with end users in the conduct of "just enough" release planning.

Should we evaluate the vendor's staffing approach?

Absolutely. Staffing approach is a key element of the technical evaluation. Evaluation criteria for an agile staffing approach should seek to understand the vendor's resource talent and ability to leverage a diverse array of development resources. Request that vendors propose a rationale for choosing the development talent and project oversight personnel based on the functionality described in the product vision (recall that FAR part 39 directs us not to prescribe any minimum experience or educational requirement for proposed contractor personnel when using PBA to acquire IT). This will help the evaluation team differentiate between agile vendors during the source selection process.

One potential pitfall is to require the designation of multiple key personnel. We recommend designating only one team member as key personnel—the project manager. When developing software, especially in a federal environment rich with legacy IT infrastructure, it is challenging to predict which coding languages and engineering techniques will be required. Key personnel designations for staff members beyond the project manager may limit the vendor's ability to source its talent pool for specific resources and expertise that can add value during the execution of a release plan.

Remember, in agile development we are buying effort in support of functionality; vendors that offer experienced developers at a fair and reasonable price are most attractive to the government because they are likely to perform better and more efficiently than inexperienced coders. This is not a rule, however. Some development tasks are more basic and may not necessitate experienced talent. Rather than prescribing the resources we think are required, we are better off instructing offerors to demonstrate why they composed the team in the manner they did. We should include instructions to the offerors requesting a description of how they will make resources available to deliver the maximum amount of functionality within the project schedule and budget constraints.

Past Performance Evaluation for Agile

Past performance is another key category for evaluating agile proposals. We recommend you focus your past performance evaluation criteria on demonstrated experience with successfully developing software using an agile development approach. Because agile development is a relatively new trend in government, it may be necessary to allow for private-sector experience to be included. Using a template that describes the key past performance areas you want to evaluate (such as nature of the work, constraints within which the vendor operated, and how it persevered in the face of project challenges and leveraged its talent pool to deliver working software) is a best practice and will help organize proposal information to facilitate the source selection process. In addition, instruct all offerors to include information about the size, scope, and complexity of their past development experience, to include its relevancy to the software being produced via your solicitation.

Cost and Price Analysis for Agile

Cost and price analysis for the acquisition of IT pose a unique challenge when evaluating agile proposals in a PBA environment. The strategy depends on contract type. For fixed price contracts, price analysis should determine fair and reasonable pricing using price competition (the comparison of proposed rates when two or more responsible offers have been received) or a comparison of rates charged to either public or private buyers for similar services. Given the relative newness of agile development in the public sector, it may be necessary to compare prices paid for agile development services in the private sector if no similar federal purchase exists.

For cost reimbursement or TM/LH contracts, cost analysis or cost realism analysis will be conducted to determine whether the cost of labor (e.g., labor rates for the developer talent) represent a fair and reasonable price to the government. In these instances, your cost evaluation strategy should focus on a comparison of the labor categories and rates proposed by the agile team. The cost analysis team should review established rate lists (GSA's IT-70 schedule is a great place to start) and conduct market research to identify average rates for development expertise and experience, especially for TM/LH evaluations. In certain instances, you may seek the assistance of personnel with specialized experience in software development who can inform your analysis of labor rates proposed or field pricing assistance if the total estimated cost of your agile acquisition is significant.[8]

Conducting a price analysis for fixed price agile proposals is a unique challenge because agile development can deliver functionality beyond what is described by the product vision. When vendors propose under a fixed price arrangement, they assume more of the risk in project delivery. This risk profile will encourage the agile development team to produce the basic functionality of each release to meet the letter of the contract, and nothing more, because any effort above and beyond the basic requirements will reduce their realized profit. This is how fixed price contracts are meant to work, and why we are hesitant to recommend them for agile development.

What evaluation strategy should be used to achieve best value?

Price must be a consideration when evaluating agile development proposals, but the focus on delivering as much functionality as possible within budget and schedule constraints makes it challenging for the acquisition team to evaluate total proposed price. If our primary constraints are schedule and budget, and the delivery of working software is contingent on the incorporation of lessons learned into the release planning process, a high-performing agile team may succeed in delivering the product vision for less than the budgeted amount. Depending on the nature of the software required, this may be sufficient and could result in an early termination of the project for the government's convenience. So how can best value be achieved when evaluating agile development proposals?

We strongly advise against the use of a lowest price, technically acceptable approach. Instead, we advocate the use of trade-offs that enable the evaluation team to consider other-than price factors such as the offeror's proposed staffing approach and specific implementation and project management techniques. Using trade-offs will allow you to make a best value determination not for delivering working software for the lowest possible investment, but for delivering the highest degree of working software within the constraints of schedule and budget.

Postaward Steps

What is the next step after the award decision is made?

Upon award, the PWS will be used to guide the development effort through creation of the product road map and release planning activities that deliver functional outcomes described in the SOO. Therefore, the first step after award is to schedule a contract kickoff meeting as soon as possible. This kickoff meeting must include the IPT, and it should be presented as an opportunity for the awardee to review its PWS with all project stakeholders.

Because agile relies on collaboration and input from end users, a kickoff session is a great opportunity to establish a collaborative mind-set early in the process. If members of the IPT have questions or concerns about the road map, they should feel free to discuss them with the development team. They can then incorporate these issues into their planning effort for the first release. In addition, a best practice when using PBA is to incorporate the winning proposal's PWS into the resulting contract, where it will serve as the product road map and guide the release planning activities necessary to continuously deliver working software.

How do you know when a release is complete?

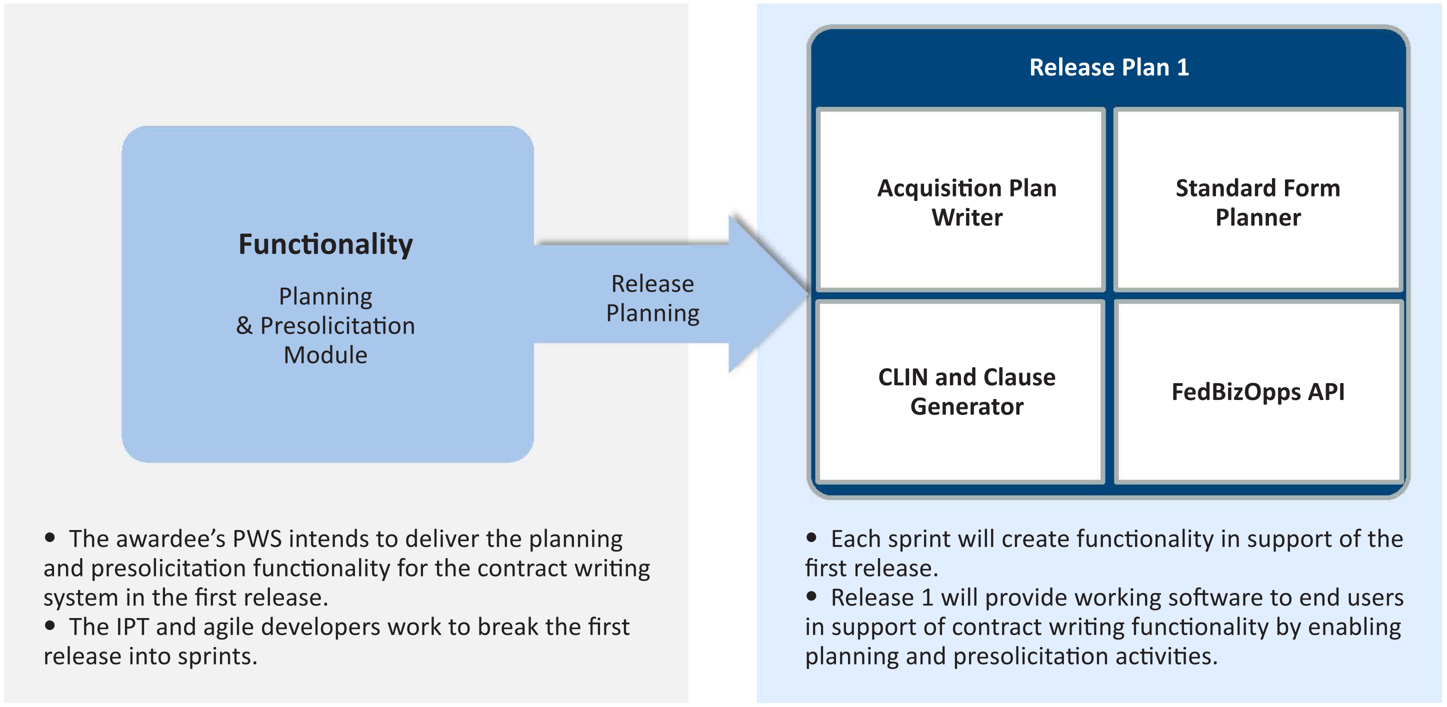
During the release planning process, the agile team will need to interview end users to determine what features of the software must be deployed through individual sprints to support performance or functionality. These interviews result in user stories that describe the functions—not the specifications—of the sprints. They also describe attributes of the working software that will dictate the user acceptance testing used to determine whether the software is ready for deployment. The user acceptance testing requirements are aggregated into an overall "definition of done" for the current release. The "definition of done" is an important tool for the acquisition professional as it helps to bind the scope of an agile procurement. Without a definition of done, it would be impossible to determine when the agile development project has achieved its contractual purpose.

When is the functionality deployed?

After completing a release, working software is demonstrated to the IPT and presented to end users for acceptance testing, providing all members of the agile development effort an opportunity to determine how the software performs. Software that passes the demonstration and testing phase is deployed to the production environment, where it is available to support the job role of the end user.

This is a fundamental element of agile development—working software is deployed within an organization throughout the life cycle so end users can benefit from the project investment in advance of contract expiration. The graphic on page 11 depicts the process of release planning and testing that occurs prior to the deployment of working software, within the context of acquiring our contract writing system example.

Execution through Release Planning


Functionality
Planning & Presolicitation Module
Release Planning
Release Plan 1
Acquisition Plan Writer
Standard Form Planner
CLIN and Clause Generator
FedBizOpps API
• The awardee's PWS intends to deliver the planning and presolicitation functionality for the contract writing system in the first release.
• The IPT and agile developers work to break the first release into sprints.
• Each sprint will create functionality in support of the first release.
• Release 1 will provide working software to end users in support of contract writing functionality by enabling planning and presolicitation activities.

How do you measure and monitor performance?

Agile development relies on empirical evidence, meaning that demonstrated functionality is of primary importance for contractor surveillance. This challenges the traditional design of performance measures and monitoring tools that rely on a set of prescribed or promised outcomes. Using service level agreements (SLAs) and other outcome-based metrics that describe functionality (e.g., usability) will increase visibility into project performance and contractual expenditures.

As we have described, agile development is not concerned with how working software is achieved, so long as it is within the constraints of schedule and budget. Progress in agile development is assessed through the team's ability to deliver working software in an efficient and timely manner. This is a concept known as "product velocity," and high performing agile teams will increase their velocity over time, meaning they become more efficient and proficient at developing software within the constraints of their project environment. There are tools and methods available for measuring product velocity; you may request that each vendor propose its preferred tool or conduct your own research to select one that works best for your need.

The exact SLAs and metrics will vary based on the nature of the development effort, so we recommend working with the agile team and your IPT to create a performance management strategy as a contractual deliverable. To get the conversation started, examples of SLAs and other agile-focused performance measures may include:

- Conduct of release planning activities
- Creation and closure of user stories through the completion of sprints
- Rate at which the product backlog is reduced (a concept known as "velocity")
- Posting of lessons learned through early releases into a collaborative environment to support software development in future releases

Working with the agile practitioners and the IPT to develop a performance management plan will ensure the acquisition professional can maintain project visibility and meet reporting expectations while not interfering with the development effort. Requesting that vendors propose a quality assurance plan as part of their proposal packages is another best practice and can serve as a further tool for technical evaluation.

Key Concept: Agile performance measures should be structured around individual releases, rather than the product vision, so they can evolve along with the working software. In this manner, outcome measures and their modalities can adapt to accommodate the progress and direction of the agile development team.

How do I know if my agency is ready for agile?

Agile requires a full commitment from the organization to be successful. Using agile terminology might make your agency feel like it is embracing agile development, but maximizing the return on an agile development project requires a full embrace of its fundamentals. Below are several questions to help you determine whether your organization is ready to use agile development. If your organization does not exhibit these traits, it may be helpful to conduct an agile readiness assessment, championed by the product owner with participation from the IPT.

- **Does my agency share information freely across component groups?**

Agile teams must have access to information such as current code and legacy software development requirements. These are primary inputs for the development team to consider as it produces working software, and lack of access to such information or legacy development requirements will hamper the agile practitioners.

If your agency has a strong collaborative environment, such as a SharePoint site or similar collaboration tool, this is a good sign your organizational culture will support agile development. If programs are constantly affected by hierarchy, stovepipes, or an unwillingness to share information across components, you should consider this a cultural "red flag" that should be addressed prior to pursuing agile development.

- **Does my agency embrace cross-functional teams?**

To get all the right players involved in an agile development project, resources from across the organization must have the support of their leadership to devote the necessary time to an agile project. If IPTs are a frequent occurrence in your acquisition environment, and are used to their full advantage, this is a sign your agency is in position to embrace the fundamentals of agile development.

If IPTs are chartered at the beginning of a project but do not provide regular support and assistance throughout the acquisition life cycle, this is a warning sign. Collaboration from all team members is fundamental to the success of an agile development project. Saying resources will be made available and actually following through on that commitment are two very different things. Team members who support an agile project, such as end users, must have the support of their supervisors to put on hold their current work to support an agile development effort. Team members also must have requisite autonomy (or authority, in the case of the product owner) to make product- and process-related decisions on a daily basis that will affect ultimate product development. This is necessary to honor the schedule and budget constraints of the agile development cycle.

- **Does my agency enable its acquisition function to work with efficiency?**

Regardless of the acquisition strategy and contract type selected to support an agile development, the acquisition team must be able to efficiently award contracts and execute contractual processes. If your agency's acquisition process is hampered by interference from third parties, inefficient secondary reviews, or undocumented practices, this can affect your ability to enable successful agile development outcomes.

While we do not advocate that acquisition professionals work with unconditional autonomy, we note that delays in the acquisition process that arise when acquisition strategies are challenged or changed after contract award will limit the benefits of agile development. If your agency's peer review process can approve contracting recommendations in the short term, this is a positive sign that your organizational culture will enable agile development. But if your acquisition function struggles to gain secondary approval for acquisition strategies that are permissible by policy and regulation, this can hinder the agile development process. If these struggles cannot be solved prior to moving forward with an agile development procurement, you likely will want to consider alternative strategies for your software development needs.

Conclusion

Agile development is a powerful philosophy for delivering successful software products within today's dynamic IT landscape. Acquisition professionals do not have to be experts in all aspects of agile development to support an agile project, but understanding its principles will enable them to structure an acquisition strategy that will enable realization of its full potential.

As agile development continues to expand through federal and private sector adoption, organizations will become more accustomed to its radically simple approach. Federal organizations that wish to take advantage of its benefits must be prepared to fully embrace its principles. While this may require a significant degree of change management to foster project management and acquisition disciplines within the organizations, agencies that blaze this trail will realize an IT environment that adapts to change and continuously delivers value through timely and useful working software. ♦

Endnotes

1 See the 25-Point Plan for IT Reform and its June 2012 Contracting Guidance to Support Modular Development; http://www.whitehouse.gov/sites/default/files/omb/procurement/guidance/modular-approaches-for-information-technology.pdf.

2 http://en.wikipedia.org/wiki/Moore's_law.

3 The Manifesto for Agile Software Development (http://agilemanifesto.org/) was signed in 2001 by 17 individuals and is considered the formal genesis of the agile movement.

4 Refer to the IBM Center for the Business of Government's "A Guide to Critical Success Factors in Agile Delivery" for a complete discussion on the concept of "just enough" planning.

5 See 16.504(c)(1)(D), Multiple Award Preference.

6 https://www.federalregister.gov/articles/2013/11/25/2013-28053/federal-acquisition-regulation-accelerated-payments-to-small-business-subcontractors.

7 See FAR 39.104.

8 See 15.404-1(e) and 15.404-2(a).

Special thanks to Dan Chenok, Paul Gorans, Greg Day, Amy Wilson, Dr. Jane Snowdon, and Dr. Simon Rakoff for their assistance in support of this *Advisory*.

The *Advisory* is published as part of the Virtual Acquisition Office™ subscription service, made available by ASI Government, 1655 North Fort Myer Drive, Suite 1000, Arlington, VA 22209, 703-253-6300, fax 703-253-6301, www.GoToVAO.com. Information and opinions are based on best available information, but their accuracy and completeness cannot be guaranteed. Layout by Julie Olver. Contents ©2014 by ASI Government. All rights reserved.