De-risking Government Technology

Federal Agency Field Guide

September 2020

GSA 18F 10x

Acknowledgements

Numerous contributors across GSA made this work possible. We'd like to thank:

Project Team

Mark Hopson

Randy Hart

Waldo Jaquith

Igor Korenfeld

Vicki McFadden

Rebecca Refoy

Alicia Rouault

With Support From

Christine Bath, 18F Design

Heather Battaglia, 18F Engineering

Kristina Britt, TTS outreach

Ron Bronson, 18F Design

Will Cahoe, 10x

Ryan Johnson, 18F Design

Nico Papafil, 10x

Steven Reilly, 18F Engineering

Dahianna Salazar Foreman, TTS outreach

18F Writing Lab

Office of Strategic Communications

Office of General Counsel

TTS Front Office

TTS Communications

TABLE OF CONTENTS

01 Introduction

Only 13% of large government software projects are successful.[1] Modern software development practices reduce that risk by delivering working code every few weeks and getting feedback from end users to ensure that the product meets their needs. Federal agencies are recognizing that their legacy development practices are risky and are shifting to this agile software development model. However, the ecosystems in place at agencies — budgeting, procurement, and oversight structures — do not support agile development practices, so our success rate remains low.

In the federal government, technology isn't the challenge — outdated practices are. This guide provides instructions to federal agencies in how to budget for, procure, and oversee software development projects, to reduce risk and wasteful spending, support teams effectively, and improve outcomes for end users.

About the editors

We work for 18F, part of the Technology Transformation Services team at the General Services Administration (GSA). We're grateful to GSA's 10x for sponsoring this work, and to the many people who contributed their time and knowledge.

If you're interested in working with 18F, contact us at inquiries18f@gsa.gov.

1 Projects valued at $6M or greater, in Europe and the United States, that were completed satisfactorily, on time, and within budget. From The Standish Group's "Haze," based on their CHAOS database.

Structure of this handbook

This handbook provides straightforward recommendations for federal agency staff to address common pitfalls in implementing modern software development practices. It is designed for anyone involved in the budgeting, contracting, or execution of an agency mission through any government program, whether it resides in a department, agency, or bureau. It assumes that taxpayer funds will need to be spent externally in some way in order to successfully accomplish whatever that mission may be.

This handbook is primarily focused on, and would benefit, any mission that requires software, although some of the material recommendations and explanations are relevant in other areas of government information technology spending, such as hardware and emerging technologies. It is divided into the three major stages for any appropriated funds spent by a government agency to further its mission.

Some portions of this handbook were written specifically as part of this broader compendium, but the vast majority of it is collected writings, produced during years of research and practice, tested by experience from dozens of people and authors. It has been compiled and edited to be practical and immediately usable by anyone who has questions on these topics, providing sufficient detail to successfully execute the principles, methods, and lessons learned from people with direct, immediate experience with them. It will be updated continuously as we advance our knowledge of budgeting to procure modern, digital services for public good.

Basic principles of modern software design

A technology project's odds of success improve when the "non-technical" government leaders who fund and oversee it understand six basics concepts of modern software development: user-centered design, agile software development, product ownership, DevOps, building with loosely coupled parts, and modular contracting. You don't have to be a technologist to understand these general concepts. Once you grasp them, it'll feel like you've gained a new super power, allowing you to cut through the jargon and technical detail, and stay focused on the basics of successfully guiding any software project.

USER-CENTERED DESIGN

All software development should be centered on the needs of the software's actual end users, the specific people who are expected to use it. These "end users" may be applicants for benefits, call center workers, case workers, other state employees or any of innumerable other groups.

Designing with and for users reduces project risks by ensuring the software is solving actual problems (as opposed to what a few stakeholders think the problems actually are). These problems are identified via a variety of research tactics, including interviews and testing for usability.

In user-centered design, all work is in the service of those end users' needs. That work is identified and prioritized in close and regular collaboration with end users, and is informed by, but not subservient to, any technical constraints. (That is, the goal of the work is to deliver value to users, which involves dealing with the realities of approved programming languages or server software, but work should never be

omitted because of the perception that technical constraints would make it impossible.) The technical team and end users regularly review the work, as it is being performed, and the development work on the new software is not considered finished until those end users agree that their needs have been met. Designing with and for users reduces project risks by ensuring the software is solving users' problems.

AGILE SOFTWARE DEVELOPMENT

Detailed, long-term plans for major, custom software projects have long been the norm in government. But, as software engineers and policy makers have learned over the years, those plans are never correct. They need a lot of costly modifications, leading to requests for more money to pay for "change orders". It's time for government executives and budget officials to stop asking for detailed long-term plans, and instead to budget for software projects in a new way.

Planning an entire project upfront is known as "waterfall" development. Imagine planning a month-long family vacation of driving around the United States. Under waterfall, this would entail planning up front each day's agenda, including the route driven, booking every hotel room, pre-paying for every meal, pre-buying tickets for admission to attractions, etc. This would never work because things change, unexpected options come up, and no rational person would want to lock in every decision at the start of the journey when they don't know what the journey holds. Instead, most people would map out the general route to be taken and plan a few major stops — the specifics would be sorted out as they progressed along the way.

"Agile software development" refers to using this trip-planning methodology for building and modernizing software systems. Instead

of relying on years of costly planning and "requirements gathering" before beginning to write actual software, agile development projects are planned only in broad strokes, with a well defined description of the overall project goal and a strong preference for *just getting started*. A small, empowered, self-motivated team (usually 5-9 people, including developers, product managers, user researchers, writers, and/or security experts) is dedicated to accomplishing that goal, using user-centered design, working in two-week cycles to deliver some actual working software.

On day one, the team plans only what they'll do for the next two weeks. (The length of a project's cycles can be as brief as one week or as long as four weeks — two weeks is the most common.) Each task they'll work on is in the form of a "user story" — a specific user need revealed by user research.[2] The entire collection of user stories to be worked on is called the "backlog."

The team works on a selected group of user stories for two weeks and, at the end, the team reviews the work that they did, tests it with end users, and then plans the next two weeks by pulling more user stories from the backlog. Repeat. Each of these two-week cycles is referred to as a "sprint."

In the beginning, the software they produce may not seem like much (and may even be replaced by something else later), but it will gradually and systematically inform the project's technical approach and help the team sensibly integrate the project into an agency's existing legacy system.

2 A user story reads in form of "as a [role], I need [this thing], so I can [accomplish this]." For example, "as a social worker, I need case notes to be cached on my phone, so that I can access case notes in areas without mobile phone service." All technical work is done in the service of addressing a user story.

Functioning software is delivered at the end of each sprint, without exception — fully-tested, fully-documented, ready to be used. In this way, value is delivered constantly, until the software is good enough to be rolled out for broad use. The team continues to work until they accomplish all of the goals or they run out of money, whichever happens first.[3]

The vendor is paid for their employees' time, not for a software system. Everything created by the vendor — software, documentation, research, designs, *everything* — is owned by government, delivered to government at the end of each sprint. Technology changes, government policies change, regulations change, laws change, and leadership's priorities change — any project that is planned in great detail up front will be unable to adapt to those changes, and will be at significant risk of failure, significant cost and deadline overruns, or costly "change orders."

By coupling agile with user-centered design, a development team can constantly iterate toward solving the needs of end users in ways that would have been impossible to learn about up front.

Earlier this year the U.S. Department of Defense's Defense Innovation Board released its Software Acquisition and Practices (SWAP) Study including a concept paper on "Detecting Agile BS," which provides a useful synopsis of agile practices, and a series of questions to help non-technical leaders understand whether those practices are being followed.

3 Stack Overflow's 2018 survey of 57,075 developers found that 85% of professional software developers use agile. And a 2015 study by Hewlett Packard found that "the vast majority of organizations [they] surveyed reported that today they primarily use Agile methods." The process described here is not extraordinary in any way.

PRODUCT OWNERSHIP

Taking back ownership of government software projects requires government teams to focus on outcomes, not outputs. This means shifting from some of the traditional Program Management Body of Knowledge practices to a product-oriented mindset.

The word "product" may sound unusual in a government context, but it's an important bit of tech lingo. "Product" is a shorthand for whatever the thing is that's being created: a website, an iOS app, an intranet application, etc. Although the word makes more sense for a business that's selling a literal product, everything else about the concept translates to government perfectly.

The product owner is the key person for any software project, and *must* be a government employee. The product owner works with users, stakeholders, technologists, and the vendor to envision the direction for the product, with an eye toward delivering value to end users as quickly as possible. They iteratively prioritize and define the work for the product team, as part of the agile process. They measure progress against clear performance indicators, and communicate with stakeholders and the cross-functional team that is building the product.

The product owner doesn't need to be a strong technologist. Instead, they should know the users of the system, the business (for example, Medicaid insurance or DMV services), and policy constraints.

A strong product owner ensures that the vision is clear, the strategy is clear, there is space for teams building the software to learn, and that they are building or buying the right thing to incrementally show value to users. They prioritize ruthlessly to ensure that the product serves user needs, and that activity and attention is focused on the highest-priority needs. They are empowered by their agency to represent

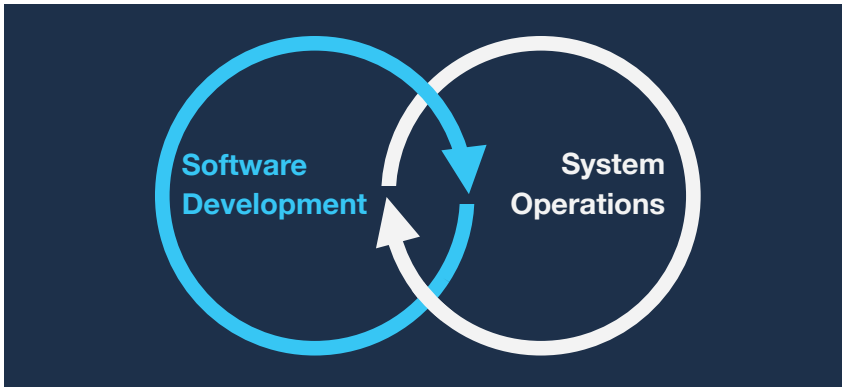
stakeholders in making rapid product decisions without the need for many layers of approval. This positioning ensures that the product owner understands everything that the development team is doing and that the needs of government are fully represented.

This is different than typical project management in government IT. The product owner won't have Gantt charts or a detailed 5-year plan. But they will have a vision for the outcomes that will be delivered to users, and have a path to executing. Their most important job is to understand what the development team is doing and to make sure it strikes the right balance between the needs of government and the needs of end users.

It's possible for a first-time product owner to learn as they go, but it's better to be trained in advance. There are many sources of agile and scrum training, some specifically for product owners. These range from YouTube video series to in-person, multi-day classes to become a "Certified Scrum Product Owner." The more important the project, the more formal and rigorous that the product owner's training should be.

DEVOPS

Historically, the teams building software have been separate from the IT teams that are responsible for operating the software once it's ready for use. A vendor might spend years building new software, and then a government IT team (or a vendor filling that role) might then require many months of work to get that software to function correctly on their servers. This is usually accompanied by frustration and finger-pointing, and can lead to project failures. To address this, government agencies often insist that the vendor building the software also host it indefinitely on the vendor's infrastructure, which has the effect of ruling out most software vendors (who are not in the



hosting business), and creating vendor lock-in with its associated high prices. Relying on these old approaches will get you less and cost more than adopting the modern software tools that are standard in the private sector.

The way to address this is with DevOps. This is the practice of coordinating the work of these two groups to automate the work that goes into testing software and moving it to a live server where people can use it — merging software development and system operations. The developers write a series of automated processes for ensuring that the software will function properly in production, over the course of writing the software itself. Developers cannot merely hand their completed work to the system operations team and declare "hey, it works for us" — they are responsible, both practically and contractually, for their code working properly.[4]

Odds are good that most of the software you use every day, whether on your phone or your computer, was written just like this. Under DevOps, testing software quality is automatic, testing software security is automatic, merging multiple developers' work is automatic,

4 For more on DevOps, see the Defense Innovation Board's "Is Your Development Environment Holding You Back? A DIB Guide for the Acquisition Community."

and moving completed software to servers is automatic. (The incorporation of security testing in DevOps is sometimes labeled as "DevSecOps.")

BUILDING WITH LOOSELY COUPLED PARTS

Large, complex software projects tend to collapse under the weight of administration. No single developer can understand the entire system that they're contributing to, yet each new member added to a project team increases the complexity of the entire team's interactions, necessitating new supervisory roles like "software architects," with whom developers must check before doing any work. The contributors need to coordinate carefully to avoid conflict between their efforts. As a team grows, they're forced to spend increasing amounts of time managing the project, and decreasing amounts of time actually doing the work.

To avoid this fate, it's smarter to break large projects into a handful of small, quasi-independent software projects. In this model, each component communicates with other components through simple, modular standards, so that any one piece can be swapped out at any time. Instead of building a monolith that everybody will lament in a few years, you build a little ecosystem, in which each piece can be upgraded and modified easily, as changing needs will demand. Each component is maintained by a single agile team, which documents the component's application programming interface (API) — the grammatical rules that other components can use to communicate with it. The teams' need to coordinate is minimal, because they can simply follow the API documentation for the other components that they need to interface with.

When each component uses abstracted APIs (think of them as common standards for using that technology), this is known as

using "service-oriented architecture" (SOA). This is the same as the concept of "interchangeable parts" that made the industrial revolution possible. Standardized couplings are the underlying concept behind cloud computing, electrical outlets, USB, Legos, trains, and countless other modern products and practices.

Building IT systems using loosely coupled parts, connected by open and available APIs, is the "magic bullet" that allows for flexible, sustainable systems that meet user needs and cost less over time.

MODULAR CONTRACTING

By combining user-centered design, agile, product ownership, DevOps, and building with loosely coupled parts, it's possible to break up a large, risky contract into a handful of smaller contracts. A contract should be small enough that the agency will have no compunction about giving no further work to a non-performing vendor, replacing them with a new vendor. (See "Procure services, not software" for how this is done.) The rest of the vendors will continue working, so the total loss of velocity will be minimal. A new vendor should have no difficulty taking over for the old one, since the old one was delivering completed, documented, tested software every two weeks. Another benefit is that small contracts may come in under your state's simplified procurement threshold, meaning that agencies can write a request for proposals, publish it, and award a contract, all within 90 days or so.

There are vendor teams that specialize in working as we've described here. As a rule of thumb, an agile development team of 5–9 people costs between $1–2M/year, depending on their geographic location.

This approach will require coordination and buy-in from your procurement teams. Procurement personnel are often accustomed

to the traditional approach of outsourcing IT projects: one large procurement based on lengthy RFP documents, asking for lengthy proposals and outdated, waterfall-style certifications and qualifications from vendors. Generally, vendors that use agile, user-centered methods don't have any idea what "CMMI" or "EVMS" is — such standards are no longer considered best practices for creating flexible and cost-effective software systems. This is a barrier to entry for many of the vendors that might be new to government and don't want to expend all of the resources required to write a proposal.

• • •

Modern software development processes are founded on user-centered design, agile software development, product ownership, DevOps, building with loosely coupled parts, and modular contracting. By understanding those core concepts, you're in a great position to understand how to **budget for**, **plan**, **and execute** software programs more effectively, and to understand the rest of these **field guides**.

Designing with and for users reduces project risks by ensuring the software is solving actual problems

02 Planning

The following sections identify challenges and strategies to mitigate risk during the planning phase of custom technology projects in government.

Assign dedicated and empowered product owners to lead development efforts

Waldo Jaquith, Peter Rowland, Miatta Myers, Vicki McFadden, Mark Hopson

CHALLENGE

Agencies are not empowering product owners to be successful.

EXECUTIVE SUMMARY

- Any Agile development effort requires an empowered, accountable, and technically proficient federal leader to succeed
- Product owners must be given the time, latitude, and authority to succeed and deliver value to end users.

RECOMMENDATION

To reduce risk of failure and enable a greater chance of success for product development efforts to succeed, the government needs to take back some of the control that it has been outsourcing. This is done by appointing a government employee to serve in the role of product owner for a development effort. The product owner will help set the team's vision and priorities, and accept the contractor's work.

The goal of a product owner is to build a product people want to use. This is different from traditional government jobs, such as project and program managers, who focus on making sure an initiative runs well, and delivers on-time or on-budget.

A common problem government custom software development projects face is that leadership has not set up the product owner to succeed.

Product ownership is often treated as “other duties as assigned,” but it is a full-time job. Product owners can’t work on several products at any given time, especially if it’s a new role for them. Their days should be filled with scrum ceremonies (sprint planning, sprint review, retro, daily stand-ups), clearing blockers for the team, attending usability sessions, meeting with users, communicating with stakeholders, and refining the backlog. Expecting someone to lead a product development team with only a fraction of their time is setting them — and the product— up for failure.

Another common problem is requiring the product owner to work through a governance board to make changes in product direction. Often, governance boards will vote on requirements, not based on any understanding of user needs, and give those requirements to product owners to execute against. That is at odds with an Agile approach. Agile methods require that product owners are empowered to act on their understanding of the end users’ needs. They should be empowered to say “no” to feature requests that do not meet user needs, and do so often, no matter who in the organization is making the request. This should be well understood and accepted. The product team should be empowered to adjust course throughout a project, based on what they learn. They should be able to communicate freely with their end users. This autonomous product team and product owner role requires cooperation with executive stakeholders. Without this autonomy and support, it is more difficult to deliver the right product to end users.

Product owners come from all walks of life. To serve in this role, they don’t need a specific education or work experience background, but they do need to understand the needs of users, be able to lead a team effectively, clear blockers, and deliver results.

They also need a willingness to learn and have the time and space to experiment, make mistakes, and grow. They should also receive training – there are many excellent scrum product owner offerings available – and would benefit from working with an experienced scrum master and/or Agile coach as they begin to master this new skill.

Involve end users early and often in software development efforts

Waldo Jaquith, Peter Rowland, Miatta Myers, Vicki McFadden, Mark Hopson

CHALLENGE

Product development efforts lack sufficient end user input.

EXECUTIVE SUMMARY

- Agile is impossible without regular and ongoing feedback from end users. Agencies can't be Agile without such feedback.
- Leadership, governance boards, or proxies should not be "deciding for the user."

RECOMMENDATION

A wide variety of users are likely to interact with government software. For government programs, an end user of a given product could differ – it could be the public, a warfighter, a field employee, or one of dozens of other groups. Today, end users are not sufficiently involved in government Agile projects. Projects that get funded are not always driven by end user needs. It's common to find "Agile" projects that lack sufficient – sometimes *any* – end user feedback.

Without regular and ongoing feedback from actual end users, an agency can't be Agile. No level of stakeholder priority-setting or requirements review board processes can substitute for active and continuous end user feedback loops. Someone who had the job of

an end user 10–20 years ago cannot serve as a proxy. Surveys do not bring sufficient user feedback into the development efforts. By not communicating with end users, teams could be solving the wrong problems. Under that approach, there is no feedback loop to validate whether the end user is satisfied, which is the primary definition of success in an Agile project. This is an extremely risky way of developing software.

Every effort should start with end user research. Every user story in the product backlog should be based on current end user needs. End users should be continuously asked for feedback on product direction to shape the product to their needs. Development teams should be empowered to make changes in direction based on the feedback of end users. In many cases, end users should be included in every sprint review, so they can see the product mature iteratively, and provide feedback on direction.

Consider tradeoffs in build-or-buy decisions, taking all factors into consideration

Waldo Jaquith, Peter Rowland, Miatta Myers, Vicki McFadden, Mark Hopson

CHALLENGE

Customizing commercial off-the-shelf (COTS) can have adverse outcomes.

EXECUTIVE SUMMARY

- Custom development versus "commercial-off-the-shelf" or COTS is often a false paradigm. Often, agencies will buy COTS and then also pay for custom software development to make the software meet their needs.
- Agencies should be wary of customizing COTS solutions to meet their needs – this software is difficult to maintain and may leave the agency locked into a long-term, sole-source relationship with the contractor.
- Most government purchases are *commercial*, not *non-commercial*, and should leverage simplified acquisition procedures.

RECOMMENDATION

Government agencies often describe challenges and the expense of customized commercial off-the-shelf (COTS) software. These efforts often start out as a pure COTS implementation, until agencies realize that they need to customize the software to meet their needs.

In these situations, the agency pays industry to develop custom software that the agency may end up locked into, especially if, as often happens, the agency did not secure sufficient data rights in its contract.

Over time, these systems become more difficult to maintain, as new features and customizations are added to the base COTS product, each of which bring it further away from actually being COTS. 18F technologists often refer to these products as "unrecognizably modified off-the-shelf" software, or "UMOTS."

Modifying COTS software[1] eliminates most of the benefits of using COTS. Customized COTS is often modified to the point where routine software updates can no longer be applied. At this point, the software requires expensive custom updates for the duration of the software's life. It also locks the agency into a long-term (and often sole-source) relationship with that contractor.

Without a path to replace highly modified COTS software or to bring it back into compatibility with developer updates, these systems require substantial maintenance expenses over time. Though COTS itself is an appealing way to gain desired functionality rapidly, the hidden costs of modification and the timeline to implement these changes may eventually outweigh these benefits.

Custom software allows agencies to build a solution that serves its unique operating environment. Custom-developed software may require more upfront investment, but over time its operational costs

1 When modifying COTS, keep in mind the existing warranty and any impacts that it may have through such an activity.

go down. When procured with due attention to acquiring sufficient data rights, custom software also eliminates vendor lock-in and facilitates no-cost reuse.

Before making a build-or-buy decision, teams need to understand end user needs, including the non-negotiable ones, so they can properly assess their options and make the best decision.

A team may want to go with a COTS solution when:

- The product is under active development and is in widespread use, which indicates that the vendor continues to deliver value and respond to market needs.
- It doesn't need customization to suit organizational needs.
- The agency will modify its existing practices to work within the limitations of the COTS software.
- The COTS software is highly mature in the marketplace, and similar organizations have successfully implemented the solution (with "success" determined by end users, not by the vendor).

A team may want to go with custom software development when:

- The COTS solution will require change requests or customization.
- The required COTS customization breaks the ability to apply routine software upgrades and patches.
- The vendor owns modifications to a COTS system.
- The COTS software comes with the baggage of having to support a suite of features that the organization will never use. Large COTS platforms are the Swiss Army knife of solutions and can leave agencies paying for large licensing costs to support unused capabilities.

- The COTS vendor is not transparent about maintenance costs for modifications, migration of existing data, ownership of agency data, or how to export agency data when the contract ends.
- No other organization has successfully implemented the COTS solution in a situation that is similar to the one the team is facing.

Commercial designations and the FAR

Agencies also mistakenly designate their development effort as *non-commercial*. This requires them to use restrictive and time-consuming parts of the Federal Acquisition Regulation (FAR), such as FAR Part 15. Most government purchases are commercial, whether they are custom software development or COTS purchases. Agencies should leverage simplified acquisition procedures whenever possible.

FAR Part 2 defines "commercial" as:

> [anything] customarily used by the general public or by non-governmental entities for purposes other than governmental purposes, and-
>
> (i) Has been sold, leased, or licensed to the general public;
>
> or (ii) Has been offered for sale, lease, or license to the general public;

This includes anything commercially available that is modified either in a way that is "customarily available" in the commercial marketplace or in a way that is:

> "...not customarily available in the commercial marketplace made to meet Federal Government requirements. Minor modifications means modifications that do not significantly alter the nongovernmental function or essential physical characteristics of an item or component, or change the purpose of a process. Factors to be considered in determining whether a modification is minor include the value and size of the modification and the comparative value and size of the final product. Dollar values and percentages may be used as guideposts, but are not conclusive evidence that a modification is minor…"

Default to open

Ian Lee, Ryan Johnson, Rebecca Refoy

CHALLENGE

Agencies are not taking advantage of the benefits of open source.

EXECUTIVE SUMMARY

Developing code in the open benefits agencies in numerous ways: improves code quality; gets active feedback from the public; makes collaboration easier among agencies, contractors, and the public; improves security; and encourages reuse.

RECOMMENDATION

18F promotes the benefits of using open source technologies and, more broadly, the value of working in the open. The open source principles underlying all our work include:

- Default to using Free and Open Source Software (FOSS). FOSS is software that does not charge users a purchase or licensing fee to modify or redistribute the source code. It contributes our improvements to that software back to the open source community.
- Develop our work in the open.
- Publish publicly all source code created or modified by 18F, whether developed in-house by government staff or through contracts negotiated by 18F.

18F asked a few of our partner agencies[2] about how open source has worked for them and what the future looks like across the executive branch:

Department of the Interior's Office of Natural Resources Revenue:
"...the principles of transparency – emblematic in both the open data and open code that power the site – continue to inform our approach. We want the site to embody openness and transparency both in the content provided and in the way we build. For that reason, we use open source software to build the site, and we use GitHub to manage our code and workflow in the open."

Department of Justice's Office of Information Policy (OIP):
"Working in the open to get active and frequent feedback from public and government stakeholders was the perfect match for the National FOIA Portal project. Given the core purpose of the FOIA, to shed light on government activities, it was very important to us to develop the first government-wide National FOIA Portal in the most transparent and open way. Working collaboratively in the open with our diverse stakeholders, we were able to create a dynamic website that meets actual user needs."

"OIP found that by working in the open, we were not only able to get immediate and active feedback from our stakeholders, but also it allowed other technologists to contribute to the overall success of the project. We would also encourage those who are unfamiliar with the open source process to approach it with an open mind."

The Department of Defense (DoD) addresses common security concerns with Open Source Software (OSS)**:**
"Hiding source code does inhibit the ability of third parties to respond to vulnerabilities (because changing software is more difficult without the source code), but this is obviously not a security advantage. In general, 'Security by Obscurity' is widely denigrated...Some OSS is very secure, while others are not; some proprietary software is very secure, while others are not. Each product must be examined on its own merits."[3]

2 https://18f.gsa.gov/2018/05/24/what-agencies-have-to-say-about-working-in-the-open/

3 https://dodcio.defense.gov/Open-Source-Software-FAQ/#OSS_and_Security.2FSoftware_Assurance.2FSystem_Assurance.2FSupply_Chain_Risk_Management

Benefits

The public funds government projects, and the government should allow the public to use what it has paid for. The public – and other agencies – should be able to leverage these investments for their own purposes. Re-use reduces redundancies across the public sector for similar investments and facilitates innovation in the private sector.

Developing work in the open allows agencies to own their code and reduces the risk of future vendor lock-in.

Building with open source technologies, and building in an open code repository, may make the final product better. By using open software and working in the open, agencies allow critical evaluation and participation from others. Inviting critique from others can be uncomfortable, but it increases the likelihood that the final product is better – bugs and vulnerabilities are found and fixed quicker and security is improved.

Open source software also shows a developer's skills. Skilled developers want to work on open source projects to demonstrate their skills to colleagues, their current employer, and future employers. When their code is available for anyone to check, their personal standards go up and other developers can contribute to the project to improve outcomes.

Require infrastructure-as-code, single-command deployment, and per-sprint government verification of functionality

Waldo Jaquith

CHALLENGE

Agencies do not own their technical stack through the use of DevOps.

EXECUTIVE SUMMARY

A great deal happens between a developer writing code and then running the code where people can use it. It is important that the government controls the entire process to avoid vendor lock-in.

RECOMMENDATION

When contractors develop software, and are also in charge of the deployment and hosting of that software, it has the potential to create a conflict of interest.

If the contractor creates an opaque and convoluted deployment process, or complex and undocumented hosting requirements, then it puts the agency at risk of another form of vendor lock-in. The code is never in the custody or control of the agency at any time, from when it is authored until it is deployed to production.

A software-development contractor should not also provide the hosting service for that software. The agency must be in charge of which software is deployed to the agency's hosting environment.

Many contractors prefer this approach, they do not want the liability associated with having direct access to government hosting infrastructure.

If the contractor properly automates the entire DevOps process and complies with the requirements of the Quality Assurance Surveillance Plan (QASP), then the agency can perform that deployment with a single command or click.

An important part of this automation process is defining infrastructure as code. That is, instead of providing prose-based instructions about the specifications for each server required ("needs one MySQL database with 2 CPUs and 4 GB of RAM"), they should provide those instructions as machine-readable code that will create virtual servers on the fly.

For example, here are instructions, written in Terraform (a programming language) that will create a web server:

```
provider "aws" {
    region = "us-east-1"
}

resource "aws_instance" "web_server" {
    ami = "ami-00068cd7555f543d5"
    instance_type = "t3a.medium"
}
```

These instructions specify the creation of a server in Amazon Web Services' Northern Virginia hosting facility, running Amazon's version of Linux, with 2 CPUs and 4 GB of RAM (what AWS calls a "t3a.medium"). A new stanza is written for each additional server that is required. This file is part of the source code developed by the vendor

and delivered at the end of each sprint. Doing this ensures that the technical infrastructure matches the needs of the software, and that government can manage its own infrastructure without giving the vendor access to the government's hosting environment.

This code can be configured to fit a project's specific needs and an agency's specific cloud provider. If a project uses Microsoft Azure instead of AWS, swap out "aws" for "azurerm". To create more, fewer, or different resources, simply make the relevant changes to the instructions.

The government can review these proposed instructions as a pull request before they are used in production, just as they would for application code. AWS, Azure, and Google Cloud are all current cloud infrastructure providers to GSA. This example is for educational purposes only and is not intended to make any statement on the relative merits of AWS or Azure.

By having contractors use DevOps and define infrastructure as code, agencies can monitor and control technical work, and retain ownership over their technical infrastructure.

Leadership should set direction and empower teams

Peter Rowland, Mark Hopson, Vicki McFadden, Miatta Myers, Mark Hopson, Waldo Jaquith

CHALLENGE

Leadership's role differs in an Agile project.

EXECUTIVE SUMMARY

Leadership's role in Agile development is to empower teams, align organizational governance and oversight practices to Agile development, and remove blockers.

RECOMMENDATION

Leadership's role in an Agile project should not be to dictate requirements, set stringent timelines, solve problems, or make decisions on behalf of the user.

Instead, leadership should support and empower the development teams, and give them permission to make small mistakes and learn from those mistakes.

Leadership has an important role to play in establishing an organizational priority to adopt Agile development, and aligning governance and oversight processes to this methodology.

The artifacts that agencies typically look at when reviewing a project's progress are time-consuming to produce and discourage

Agile development. "Waterfall muscle memory" is strong in most agencies; most of the internal practices and processes have been established and reinforced over decades, and are not suitable for Agile development. Employees who have spent years performing work in the service of leadership's needs will need time to learn how to prioritize user needs.

Internal management processes that demand detailed requirements documentation will make Agile impossible. This is a large shift that will take a long period of time, and missteps will be made along the way. It's essential to have leadership in place to spearhead this effort. Without a significant and focused effort to shift an organization's culture, Agile development is unlikely to succeed on a larger scale.

Other constructive involvement of leadership and key stakeholders in Agile development may take the form of providing funding, helping to set the product vision, monitoring product roadmaps and backlog, attending sprint reviews and user research sessions, helping overcome bureaucratic hurdles, and serving as authorizing officials in security accreditations.

The level of leadership and stakeholder involvement needed is personality- and team-dependent, so there likely isn't a one-size-fits-all approach that will work with all teams.

Software development efforts should be tightly scoped to reduce risk and avoid overspending

Mark Hopson, Vicki McFadden

CHALLENGE

Programs are receiving too much money for software development projects.

EXECUTIVE SUMMARY

- Building technology with loosely coupled parts means that agencies no longer need to undergo the big, upfront design, procurement, and implementation of large systems. Systems should be split into modules that deliver functionality to users, and the infrastructure to support a given module should be determined by the development team in the sprint in which it is needed.
- Scope projects so that one or two scrum teams can deliver functionality to users, keep the period of performance to three years or less, and don't spend more than $10 million per modular contract.

RECOMMENDATION

18F often works with a program that intends to spend between tens and hundreds of millions of dollars for a software development project.

When we ask why they're spending so much, we often hear that the project is important to the agency's mission, the project is a priority for agency leadership, or that their end users are really counting on them to deliver this functionality.

It's counterintuitive, but spending *more money on a project increases the chances of failure*. Of government software projects that cost more than $6 million, only 13% succeed. But of those that cost less than $1 million, 57% succeed.[4]

Old trap: Fund and do all the things, right now

Here's a common government IT project story.

> A program is identified as a priority. It is given too much money. The agency spends months or years gathering requirements from every stakeholder imaginable. The scope becomes untenable. The contract is awarded, with the perception that the risk has been outsourced to the contractor. Leadership celebrates. Government gets red/yellow/green "traffic light charts" monthly that show that the project is on track. The agency cycles through several project managers, each playing "hot potato" with this pending disaster. Several years go by. The traffic light charts finally shift to yellow or red on cost, schedule, performance, or all three. Little to no new functionality has been delivered to end users. Finger-pointing begins.

New approach: Pick a place and start delivering value

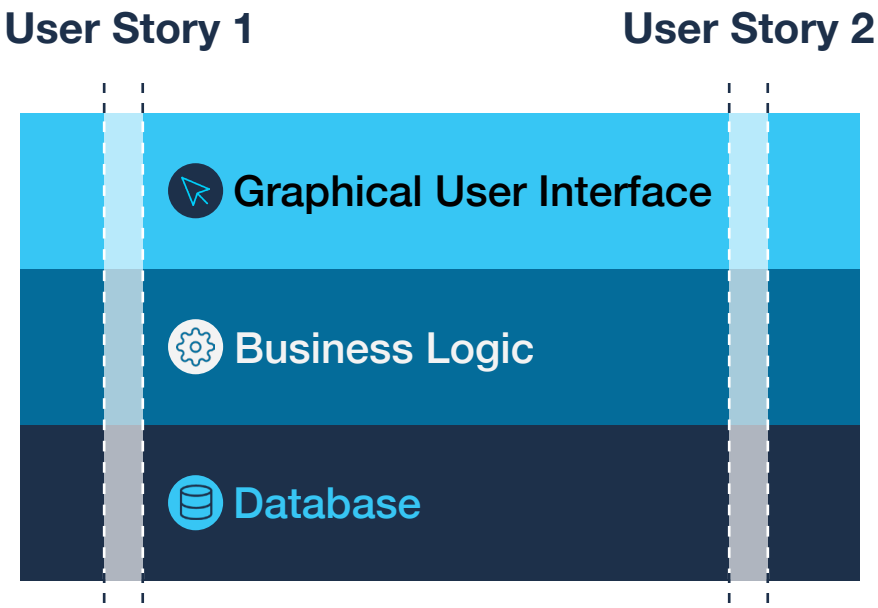
Instead of this tried-and-true approach that nearly guarantees failure, we advocate for breaking up large, monolithic systems or requirements into discrete chunks that can be delivered by one or two scrum teams.

4 Projects valued at $6M or greater, in Europe and the United States, that were completed satisfactorily, on time, and within budget. "Haze Report," The Standish Group, 2015.

A few rules:

1. **No big up-front design**: Under waterfall development, agencies spend months or years getting infrastructure procured and deployed so software development can happen. But this becomes trivial with cloud hosting — instead of spending a lot of time figuring out infrastructure, database, business logic, user interface, etc., those can be created by the scrum team as they work on user stories as "slices" that cut across these layers.

2. **No big-bang launch**: Many times, agencies say they intend to swap out a legacy system for a new system all at once, usually described as happening in a single day or over a weekend. This is rarely needed. Most often, we suggest they use a "strangler" (also known as "encasement") strategy to modernize legacy systems.



3. **Build an ecosystem**: Don't replace the hated old monolith with a hated new monolith. Instead, build a little ecosystem of services, each coupled to the others via open application programming interfaces (APIs). This architecture is the premise of Service Oriented Architecture (SOA). It allows projects to be broken into modules that require minimal-to-no coordination between the teams working on each module.

Once agencies understand that the new system does not need to be entirely designed or defined up-front, we can move directly to how to start work and start to deliver value to users. Usually we help agencies by charting their desired functionality on a 2×2 chart, with user value on one axis and technical complexity on the other.

We start work on the functionality that has the highest user value and the lowest technical complexity. While we want to quickly deliver value to end users, we don't necessarily want to pick the most technically challenging part of the project to start with. Some teams want to go after the most technically complex chunk of work first as a form of pathfinding, but that raises the stakes in a way that inhibits adoption of Agile. We're not dogmatic about what "chunk" of work gets picked first, just that the work is tightly scoped and will allow the team to deliver value to users relatively quickly.

This "chunk" of the overall system should be assigned a product owner and allowed to write and award their own contract.

At this point, agencies often face two challenges:

1. **Move to Agile by team**. Moving to Agile tends to bring out organizational antibodies that stifles and discourages innovation. To combat this, we start small. Allow one or two brave teams to test out Agile, learn, and recommend for what procedures, practices, and tools must be tweaked to enable Agile for the entire organization.

If several teams are told to adopt Agile at the same time, that does not allow for this important learning to occur and scale naturally. A large-scale, simultaneous Agile transition requires every team to undergo the same painful and frustrating learning process. These teams will probably develop their own solutions that will be based on the path of least resistance, rather than what is right for the enterprise.

2. **Don't consolidate contracts**. There is no "economy of scale" when buying professional services. There is no reason to combine several product teams' needs into a single contract award. It does not reduce complexity — rather, it hides the complexity inherent in any development effort. Consolidating contracts increases risk by creating a single point of failure. And it does not allow the agency to buy specialized skills that it might need for just one team.

Pricing the cost of an Agile software development team

The ideal size of a scrum team is somewhere between four to nine people. Let's pick the high end, nine people, and put together a reasonable mix of talent for the team. Using GSA's Contract-Awarded Labor Category tool, it's simple to estimate the labor cost per hour and multiply that by a reasonable number of hours that a person would be expected to work in a given year: 1,880 (accounting for holidays and leave). Our independent government cost estimate (IGCE) in this scenario, for one year of performance, is $1.9 million.

Note: The following table represents an estimate using the industry average cost of purchasing services from a private company offering Agile software development. If an agency requires their development teams to be on-site or have top-secret security clearances, the pool of available companies to compete on this work will be drastically reduced, and the agency can expect the average hourly labor rates to be substantially higher (especially as the clearance process itself is an added cost). Also, these positions and their hourly rates vary greatly depending on location within the United States that the employee is located.

Position Title	Number of people	Average hourly labor rate[5]	Cost
Software Developer Lead	1	$133	$250,040
Design Lead	1	$123	$231,240
Software Developer	3	$113	$637,320
Designer	3	$102	$575,280
Content strategist	1	$144	$270,720
Total			**$1,964,600**

Generally, we advise keeping contracts under $10 million for the entire period of performance, which we usually set at three years (one base and two one-year option periods). An Agile software development team will produce a great deal of user value during that time. If the work is scoped appropriately, the team should be able to deliver all the major features that users need.

Software is never "done." Even after three years of performance, an agency should expect to award a contract to another, perhaps smaller, software development team to continue enhancing the software and fixing user issues. If not, user needs and technical needs will continue to change, while the software fails to change to accommodate those needs. In the eyes of the end users, the latest and greatest software will slowly deteriorate to become the hated legacy software.

5 The CALC tool provides a range of Multiple Award Schedules (MAS) ceiling prices for a given labor category. Decisions on reasonable comparison pricing for labor is the judgment call of the contracting officer. The average hourly labor rate will fluctuate over time based on contract awards.

Communicating cost, schedule, and performance to stakeholders

In the bad old days of software development, an agency painfully detailed out all conceivable requirements, and industry provided a cost and schedule in their bid. This gave the false impression of certainty, because projects rarely ended on time, or on budget, or delivered the functionality that was desired. More often, there were many modifications to the contract to add requirements over time, and the price skyrocketed and the schedule slipped.

With Agile development, an agency can give a reasonable estimate for cost and schedule, but the performance is what will change over time based on user research and iterative development.

Cost is calculated by using the labor rates proposed by the winning vendor.

Spending more money on a project increases the chances of failure

Schedule is the entire period of performance or some subset of that. For example, some agencies want to deliver a minimally viable product (MVP) in the first 9 months of a software development effort. That's great: it gives the Agile software development team a timeboxed window to complete a useful but limited set of functionality for users. But the agency should know that the product will not be done at this time, and will continue to get refined and have new features added post-MVP.

Performance is unknown at the start of the project. The agency will have a product vision and a sense of the outcomes they're trying to create for end users, but specific functionality — and the order in which that functionality will be delivered — is unknown and unknowable. Over time, however, an Agile software team should be able to provide some estimates about when certain functionality should be delivered.

Product owners should develop a product roadmap and share it with stakeholders. Agencies should not use product roadmaps to forecast what will be delivered when, but use them to help communicate the priority order in which work is anticipated to be done. They are designed to help deliver the highest value to users within a given budget or time constraint.

Clear ‘path to production’ before awarding a contact

Heather Battaglia, T Carter Baxter, Kelsey Foley, Mark Hopson, Waldo Jaquith, Vicki McFadden, Steven Reilly, Greg Walker

CHALLENGE

Not having a clear path to production derails development efforts.

EXECUTIVE SUMMARY

It is demoralizing and expensive if a contractor is brought on to develop software and they can’t access the hosting or deployment environments for months. Make sure the path to production is clear from bureaucratic obstacles and well-documented before awarding a contract for Agile software development services.

RECOMMENDATION

Federal agencies often assume a clear path to production. However, when the work with a contractor gets started, delivering and deploying code becomes a challenge that creates a lot of wasted time, effort, and frustration. This can quickly escalate, creating a lot of tension and ill-will at the start of a project

Before a contract is awarded and the contractor team starts working, the agency should validate whether software can be deployed to an environment where users can see the work. There are two options: prototype a solution, or ask questions to gain clarity before a contractor gets started. Ideally, do both.

Prototype a solution

Conduct a short prototyping process (as simple as a single-page “Hello, World” website) to test your agency’s ability to support an Agile software development project in terms of technology, human resources, and policy. The prototype is a disposable artifact of the process, not something that will ever be deployed for public use.

This work will help the team understand their tool preferences and document internal processes. This will likely be useful supporting documentation in a Request for Proposal (RFP) or to give to the winning contractor so they can get started quickly.

The prototype should answer these questions:

- What is the administrative process to gain access to the hosting and deployment environment?
- What processes or policies do a software team need to work through to provision services and deploy applications to them?
- Which stakeholders are required for approvals?? What things do they need to approve? What form do applications for approval take?

Ask detailed questions

Pull together a meeting with the relevant individuals at the partner agency — technical, security, Continuous Integration/Continuous Deployment (CI/CD), etc. — to get clarity on the following questions.

- How are existing products hosted and deployed? Who is involved in those processes?

- How do we get access to the agency's deployment environment (e.g. Cloud.gov, Amazon Web Services, Microsoft Azure, on-premise servers)?

- Are there existing technology stacks, solutions, or components that are approved for use, or recommended? Are there any strong preferences among those options?

Having adequate answers to these questions is the *minimum* that an agency should have going into a build. If they don't know the answer to these questions, or the answers are murky, more investigation is needed prior to publishing an RFP.

Success criteria

These are the general criteria to ensure an agency is ready for an Agile software team, and can continue with contract award:

- There is access to a hosting environment, which somebody at the agency administers.

- There is an organizational account on a social code repository (e.g., GitHub, GitLab, or Bitbucket) for the agency, administered by one or more employees of the agency.

- There is a process by which changes made to code on the repository are automatically deployed to the hosting environment, and the agency has the ability to release frequently (a.k.a, DevOps, or CI/ CD).

If these success criteria are known, a team can feel confident that they can award an Agile software development contract and that the contractor onboarding process should be relatively smooth.

Give teams access to the remote collaboration tools that they need to be successful

Peter Rowland, Randy Hart, Mark Hopson, Waldo Jaquith, Vicki McFadden, Miatta Myers

CHALLENGE

Distributed teamwork is impeded by restricted access to collaboration tools.

EXECUTIVE SUMMARY

Remote collaboration is incredibly difficult for teams if they are not given access to the necessary collaboration tools they need. Unfortunately, this is commonplace in government.

RECOMMENDATION

Distributed teams are normal at agencies, as are distributed teams of contractors performing software development work. Allowing employees to work remotely is a great recruitment tool and attracts the best and brightest. It's also more cost effective, increases diversity, increases resilience (for continuity of operations planning), and improves work/life balance for teams. Allowing contractors to be located in a different location than your headquarters increases competition, unlocks the best development resources, and saves money. There's a 150% difference in the salary of software developers between the most-expensive and least-expensive states in the United States. There's even a similar salary range *within* some states, such as between Seattle and Spokane, Austin and Abilene, Los Angeles and Eureka.

However, remote collaboration can be difficult for agencies. Agency network restrictions and software approval policies make collaboration with remote team members difficult. Many commonly used tools for video-conferencing are blocked in government buildings. Federal employees often resort to workarounds, like using their personal mobile phones, to get access to the services they need.

Without a set of commonly available collaboration tools, especially for video-conferencing, it is much more difficult for agency teams and contractors to practice Agile development, where frequent communication and feedback are needed to facilitate quick decision-making and prioritization of tasks. The product owner should always know what the team is working on, and the team should be readily available to answer questions or huddle to problem-solve issues as they arise. Every sprint, there should be a demonstration of functioning software. Working like this via phone calls, emails, and PowerPoint presentations is *incredibly – and needlessly – limiting* for a distributed Agile team.

In the past, our teams have used the following tools to support remote work. See this article to learn more about managing teams remotely. Some of the tools 18F uses:

- A virtual room like Google Hangouts or Appear.in
- Mural or a similar sticky-note tool
- A collaborative writing and editing tool like Google Docs
- A project planning tool like Trello, Jira, or ZenHub
- Code repositories like GitHub, Bucket, etc.

Agencies should determine which collaboration tools their teams need and make those available. As an interim step, agencies may want to develop a provisional Authorization to Operate (ATO) process for piloting tools that are relatively low risk before figuring out which tools should go through the ATO process to be rolled out more broadly.

Invest in technology incrementally and budget for risk mitigation prototyping

Mark Hopson with special thanks to Charles Tetreault and other contributors.

CHALLENGE

Budgeting for large, risky investments in major software programs years in advance increases risk of failure during implementation.

EXECUTIVE SUMMARY

- Compiling several smaller software development projects into a single large project might seem easier — one budget request and one vendor contract — but it increases the risk of failure.
- Instead of a “lift and shift” approach to planning costs, use a risk mitigation budgeting strategy to incrementally discover needs, gather information, and increase the likelihood of successful modernization.

RECOMMENDATION

Instead of making large, risky investments in major software programs years in advance, agencies should apply Agile principles to break up those larger, monolithic projects into smaller, incremental budget allocations.

This idea of modularity can be found in the Clinger-Cohen Act of 1996 as enshrined in FAR Part 39 - Acquisition of Information Technology (which introduced modular contracting).

Taking a modular approach compartmentalizes failures, reduces the risk of failure, and brings the size of projects below the threshold for greater agency oversight.

The federal budgeting process is a sequential progression of estimates, documents, and revisions between the executive and legislative branches. The executive branch can only request funds from the legislative branch to carry out an agency mission.

This request begins with the submission of an estimated budget to OMB as part of the preparation of the annual President's Budget. This process is extensively detailed by OMB's policy memorandum for the Preparation, Submission, and Execution of the Budget.[6]

When the budget is submitted to the legislative branch, both chambers of Congress undertake an extensive review and approval process. Ultimately, this lengthy process results in the passage of a law whereby Congress grants appropriations to the executive agencies to carry out their respective missions. An appropriation is the authority to incur obligations and make payments out of the Treasury for specific purposes. This power to appropriate lies solely with the legislative branch, known as the "power of the purse," from Article I of the U.S. Constitution.[7]

6 The A-11 memo section related to Information Technology can be found here.

7 As a way to enforce its power of the purse on the executive, there is a significant body of law, called fiscal law, that controls how agencies may obligate Congressional appropriations. Examples include: the Purpose statute (31 U.S.C. § 1301(a)), Antideficiency Act (31 U.S.C. § 1341), Adequacy of Appropriations Act (41 USC 11), Miscellaneous Receipts Act (31 U.S.C. § 3302(b)), *Bona Fide* Needs (whose statutory basis is found at 31 U.S.C. § 1502, but is defined and interpreted through numerous GAO opinions), and Account Closing (31 U.S.C. § 1555). While readers should consult with their General Counsel on specific application or interpretation of laws, these topics are worth highlighting potential issues they present for modern software acquisitions in agencies.

Almost certainly[8], there is a many-month delay for government agencies receiving its new year funding, called a continuing resolution (CR). With a CR, agencies can spend only at the levels they were authorized in the previous year. New projects can't start. Many-month CRs stall contract awards and the agency's ability to start new modernization projects.

Receiving an appropriation enables an agency to incur an obligation, which is an action that creates a legal liability or a definite commitment on the part of the government for the payment of goods and services ordered or received. A purchasing contract, in other words. The biggest complication created from this process is that it begins about 18 months before receiving the appropriation.

This kind of planning is counter to the adaptive, Agile development, which boasts a vastly greater success rate for building high quality, user-centered digital products and services.

Suzette Kent, former Federal CIO, has said, "We still fund in single-year increments. We still fund like a project has a finite start and stop date. It does not. Just like we have to continually fund the refurbishments of our national parks or our roads, our technology infrastructure is no different."

Another problem of the current budgeting process is how appropriations are categorized. Software is considered an "asset" that exists in one of three life cycle phases:

- Development, Modernization, Enhancement (DME)
- Operations and Maintenance (O&M)
- End of Life (which can come in many different forms such as being eliminated, retired, consolidated, etc.)

8 Since 1977, the first year under the current system, Congress has passed appropriation bills on-time 4 times: 1977, 1989, 1995, 1997.

DME spending refers to agency projects that results in a "new" IT asset, or modifying an existing IT asset to "substantively" improve its capability or performance.

O&M spending, sometimes called "steady state," refers to the expenses required to operate and maintain an IT asset in a production environment.

This is important because an agency generally cannot spend appropriated funds intended for one category on a different category.

While civilian agencies have to choose between one of two categories, DME or O&M, defense and intelligence agencies have the added frustration of another category, Research, Development, Testing, and Evaluation (RDTE).[9]

To complicate things even more, many agencies already have software systems in place to satisfy information technology needs with corresponding budgets and contracts.

How does an agency make all of the necessary changes to the existing system while at the same time planning and executing for the unknowable future?

This presents many challenges for agencies. For example, agencies that want to modernize their systems have to write and award a new contract for these DME-funded improvements. The winning contractor will have to work with the existing, O&M-funded contractor to implement those improvements.

9 https://www.dau.edu/Lists/Events/Attachments/106/06-27-2018%20Color%20of%20Money_GMartin1.pdf

These two contractors may have different contracts, different working styles, and no incentive to work together effectively. It often leads to a lot of finger-pointing and frustration.

To avoid the extra effort of requesting funding, awarding, and managing a new contract for these improvements, many agencies simply modify their existing O&M contract for needed enhancements, and hope the cost is low enough so no one notices – or cares – that they agency should have made a DME request.

A single agency system may combine annual software licenses, software-as-a-service tools, and custom code. Hiding this complexity with a single contract / vendor makes untangling knots and making improvements very difficult.

Breaking the Legacy Cycle

According to one GAO report on the current state of software budgeting in the federal government, "in several situations...agencies are not sure whether to report costs as O&M or DME" and so "agencies default to reporting as O&M."[10]

In fact, most agencies spend less than 20% of their annual budgets on DME, meaning they seem to be trapped in perpetual O&M spending. One major reason for this, as GAO and OMB cite, is that "agencies tend to categorize investments as O&M because they attract less oversight, require reduced documentation, and have a lower risk of losing funding."[11] This last point is especially telling.

If an agency gets their submitted budget estimates wrong, they can request approval from OMB and then from Congress to transfer

10 https://www.gao.gov/assets/680/677574.pdf at 18

11 https://www.gao.gov/assets/680/677574.pdf at 18

amounts among these different categories. However, this can be a very lengthy process and getting approval isn't guaranteed. If the funds have to be obligated in the same fiscal year they were appropriated, they may run out of time.

This logistical issue has resulted in a steady increase in spending on O&M for legacy systems. In 2010, agencies' spending on O&M was 68% of the federal IT budget. By 2017, the amount of spending on O&M was at 77%.[12]

An agency knows the least about what it will take to fully modernize a legacy system when they draft a budget for that work.

Risk Mitigation Prototyping

Consider a hypothetical production system, meaning actively in-use, for a critical agency mission need. This kind of system would very likely be classified as a "major IT investment" for budgeting purposes. It's not uncommon for these kinds of systems to cost upwards of $20 million a year in O&M cost, based on publicly available information searchable on ITDashboard.gov.[13]

The riskiest way that an agency could try to make this transition would be to take that same $20 million spent on O&M for the existing system and simply transfer it into a request for DME funds, coupled with a monolithic contract.

12 https://www.gao.gov/assets/680/677574.pdf at 13

13 ITdashboard.gov provides federal agencies and the public with the ability to view details of federal information technology (IT) investments online and to track their progress over time. The IT Dashboard displays data received from agency IT Portfolio and Business Case reports, including general information on over 7,000 Federal IT investments and detailed data for over 700 of those investments that agencies classify as "major." According to data collected for the current 2020 fiscal year, there are 514 investments being tracked on the dashboard. Of those, 277, equaling 53.9% are at medium risk, and 32, equaling 6.2% are at high risk.

In our experience, risk mitigation prototyping provides far greater insight to the potential pitfalls, stumbling blocks, and other concerns that are only discoverable by actually working with the software code of a system. Risk mitigation prototyping results in code that can be deployed into production for end users or as a useful artifact in the solicitation process. Often, hurdles to deploy to production are discovered that should be mediated before awarding a modernization contract.

In our work with agency partners at 18F, we can often accomplish some form of risk prototyping even with a small team of 3-4 people from GSA. Just a few sprints' worth of effort can reveal information that would never have been known without a substantially riskier investment.

We often include our risk mitigation prototypes with any eventual solicitation to industry. In our experience, potential bidders find this kind of artifact valuable in both deciding whether they would be a good fit for the agency's needs and deciding how to staff a team for such an effort.

If agencies are unable to get budget requests for prototyping effort, there's an excellent alternative through the Technology Modernization Fund (TMF) housed within GSA.

Change Agile software development appropriations

The Defense Innovation Board's (DIB) report Software is Never Done: Refactoring the Acquisition Code for Competitive Advantage made a number of recommendations to try and improve the success rate for defense agency software projects. In that report, the DIB makes the case that:

> To undertake this path, Congress and OSD must write new statutes and regulations for software, providing increased (and automation-enabled) insight to reduce the risk of slow, costly, and overgrown programs and enabling rapid deployment and continuous improvement of software to the field. Laws will have to be changed, and management and oversight will have to be reinvented, focusing on different measures and a quicker cadence.

There's a promising new development on this front to reduce the complexity and challenges created by multiple types of appropriations for software. This proposal in the DoD would create a single type of appropriation to use for software needs, regardless of whether it captures what has historically been bucketed under RDTE, DME, or O&M. This single type of appropriation would resolve a host of issues caused by the appropriation categories and definitions. It would likely be welcomed by many operational components trying to deal with the complexity of modern software in government.

An agency knows the least about what it will take to fully modernize a legacy system when they draft a budget for that work.

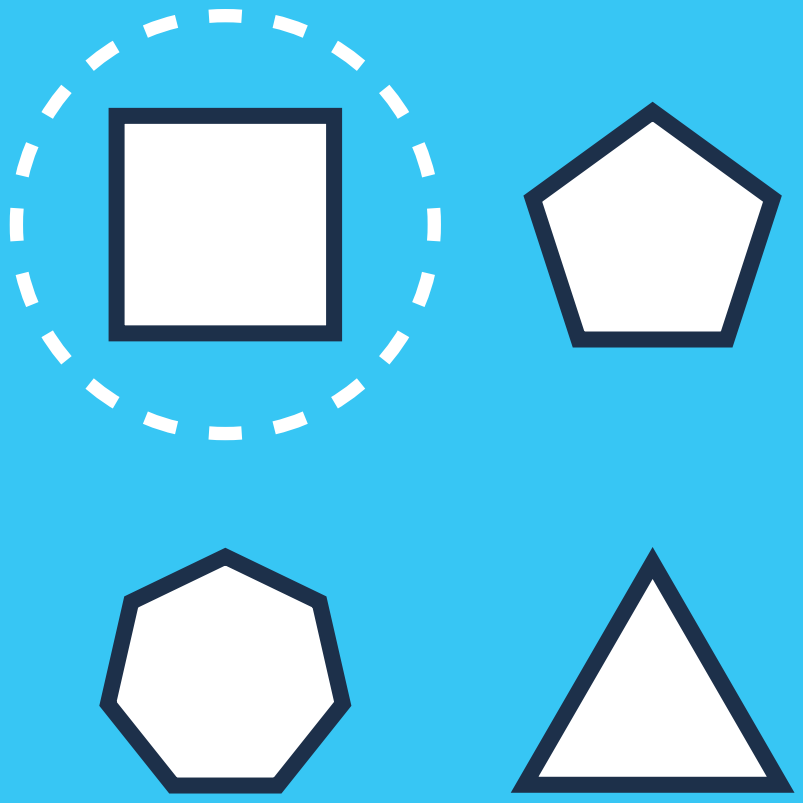
In the meantime, when budgeting for what a future state may look like, the best bet for agencies to make is to start small. Through a mechanism like risk mitigation prototyping, agencies can effectively create a form of risk mitigation budgeting that is evidence-based and incremental. It is in the original spirit of modularity conceived of in the mid-1990s, when the Clinger-Cohen Act was passed to help control "system development risks, better manage technology spending, and succeed in achieving real, measurable improvements in agency performance."[14]

The first meaningful step to provide the federal government with the same world-class technology available commercially is to address these risks as early as possible in the process to reduce the likelihood of failure.

14 https://www.gao.gov/products/GAO-11-634 at 1.

03 Deciding what to buy

The following sections identify challenges and strategies to mitigate risk during the research and solicitation phase of custom technology projects in government.



Conduct Modern Market Research

Mark Hopson

CHALLENGE

Agencies do not use industry best practices for market research.

EXECUTIVE SUMMARY

- Market research is vitally important to ensure successful outcomes with any acquisition. Given the dynamic nature of software, this is especially true.

- If done well, market research can provide reliable, accurate information that will help shape requirements, competitors, and ultimately the final product. If done poorly, then the project will be plagued from the beginning, which could lead to delayed schedules, increased costs, and ultimately, dissatisfied users. More often than not, many of the major failures involving government IT projects started off on the wrong foot based on the conclusions reached in market research.

RECOMMENDATION

What is market research?

Market research is defined as “collecting and analyzing information about capabilities within the market to satisfy agency needs.”[1] FAR Part 10, the section of the FAR for conducting market research, is only two pages. It provides extraordinarily broad discretion for how

1 FAR Part 2.101. The concept of an agency’s “needs” is used interchangeably by many agency personnel with the term “requirements.”

agencies may conduct market research. Because of such discretion, many agencies rely on outdated ways to learn about companies, technologies, and broader trends that may influence a government technology program.

Market research is a continuous, dynamic process of determining, collecting, and analyzing the availability of commercial goods and services. This ever-changing state requires keeping up-to-date with changes in the marketplace.

A government employee can easily spend one third of their time educating themselves about a good or service they may eventually acquire.

For this approach, market research has two distinct methods: market surveillance and market investigation:

Market surveillance is an ongoing process to stay informed on industry trends, new technologies, and general information about the marketplace of offerings for the goods and services needed to fulfill the mission.

Market investigation is a comprehensive, focused process on specific sources, materials, or potential competitors to fulfill a requirement usually done in order to complete a market research report for an active procurement.

The easiest way to keep the two clear is to think of market surveillance as being strategic in nature, whereas market investigation is more tactical.

For example, consider something that is very popular *because* it is essential for digital products and services: user experience (UX).

Market surveillance would involve a series of broader questions such as:

- What exactly is user experience?
- What does the practice of user experience design consist of?
- What makes for a good user experience?
- What makes for a good user experience provider?
- What kind of qualifications and experience do good user experience providers have?

With **market investigation** the questions become more pointed:

- Who is a good user experience provider?
- Where can I find a good user experience provider?
  - Are there professional associations or conferences for user experience?
  - Are there any trade publications or other information sources about user experience?
- What work have they done before? Have they done any work for government agencies before?
- Do any of these companies have existing contracts through an available Federal Supply Schedule (FSS) through GSA or some other Governmentwide Acquisition Contract (GWAC)?
- Are any of them under a recognized socioeconomic program or status such as the 8(a) program or Service-Disabled Veteran-Owned Small Business (SDVSOB)?

By considering market surveillance and market investigation separately, the advantages become apparent. By consistently conducting market surveillance, a government employee has a baseline level of information that is accurate, relevant, and timely. An agency can dive into specifics (market investigation) whenever

the agency need arises. This makes it far easier to complete necessary tasks like writing a market research report or an acquisition plan, or even conducting a competition.

When considering an agency's potential software needs, good market research is crucial to determine whether or not commercial off-the-shelf (COTS) will suffice or whether a custom Agile development effort is needed.

Often, an effort's success or failure is based on a decision made in this early step before any solicitation is issued or actual development takes place. If an agency makes the wrong decision based on poor information gathered in this step, it often leads to what 18F refers to as "unrecognizably modified off-the-shelf" (UMOTS).

How to conduct market research

Everyone is a consumer. People buy all kinds of stuff in their personal lives. However government officials have to take different considerations into account when acting as stewards of taxpayer dollars. Agency staff must consider themselves as a public's buyer. Agencies must do rigorous research across numerous sources to gather enough of the right information.

Keep these points in mind when you are conducting market research:

- Market research is meant to be a forecasting exercise, not a legally binding process. Market research cannot be used in place of the government's source selection or evaluation process to determine a contract award, nor can it favor a specific vendor or company that will eventually be awarded a contract. It's also very difficult for people to keep themselves from jumping to this step (we're all human). Keep those two procedures (surveillance and investigation) separate. Don't try to rush right into picking something when the real goal is to just see what is available.

- Market research shouldn't decide a preferred or specific manufacturer, model, or brand. This is prescribed in FAR Part 11.104 for a not-so-obvious reason: honing into a specific manufacturer, model, or brand eliminates all of a buyer's negotiating leverage.

  This is one of the less appreciated reasons for the legal mandate to compete an agency's requirements to the maximum extent practicable. It's to ensure you don't lose bargaining power.

- Expect things to change. Agencies will rarely know the exact source to satisfy their requirements up-front. That's ok. Often needs will change by the end of a market research process, which will impact acquisition strategy and/or contract vehicle.

One big mistake people make when it comes to government acquisitions is thinking of every single step as a strictly linear, sequential process. If the process is mapped out, it's a common mistake to assume that market surveillance always comes first, followed by market investigation.

But this isn't really the case in practice. A good market researcher will conduct both market surveillance and market investigation as the mission needs dictate. Usually they occur in parallel, because one will inform the other and vice versa. To do that well, agencies need to think about how to generally gather reliable information regardless of the formal title for any given step.

In the Evaluate contractor proposals based on industry best practices section of this handbook, we provide a list of what to look

for in a quality vendor proposal. If you are conducting market research to acquire this kind of developer team from a company, then you can use those same questions during market research. These questions are useful in the context of evaluating a competitive bid, but they can *also be used* to help you do your agency's information gathering and market research.

By the time an agency enters into a legally binding competitive bidding process, they'll be prepared and equipped with a much clearer understanding of what matters.

Sources of Market Information

Market research isn't really different from any other kind of research, so some basic concepts to consider about gathering information are "sources" and "methods."

Sources of information is a fairly straightforward concept. For buyers, it could mean going directly to a seller with pointed questions about some product. However, a government employee needs to account for bias and the reliability of the information they're getting from that seller. A single source rarely provides the most comprehensive understanding of anything.

Here's a layout for someone acting as a buyer for the American public.

As a best practice, government buyers should rely on multiple primary sources as well as secondary sources.

Primary

- Vendors
  - Manufacturers
  - Distributors
  - Resellers
- Other buyers
  - Private sector
  - Other federal agencies
  - Colleagues
  - State governments
  - Non-profit organizations
- Independents
  - Experts
  - Specialized consultants
  - Research companies

Secondary

- White papers or similar position statements
- Trade journals
- News reports
- Academia
- Subject-matter literature
- Databases
- Case studies

To put this into context, consider a major purchase in someone's personal life – buying a car.

Unfortunately, given the way that many government employees conduct market research and government contracting would be like someone walking into a single dealership they happened to be near, go to a single car model on the showroom floor, and pay the posted price for that car on the window immediately.

No one does that when they buy a car with their own money.

Cars are expensive, and reasonable people will do some form of market research before they buy. They will compare the available options of dealers, models, warranties, and all of the various factors to consider.

It should be the same when you are acting as the public's buyer.

Consider any typical exchange between a buyer and seller:

> **Buyer:** Does your company offer X?
>
> **Seller:** Yes of course we do. We sell the best X out there.
>
> **Buyer**: Great, I am interested in purchasing it.
>
> **Seller**: You should definitely buy mine over anyone else's out there.
>
> **Buyer**: Why?
>
> **Seller**: Mine is the best!

How do government employees know if the seller's assertions here are true and accurate? They can't know, not without other information to compare it against.

Methods for Conducting Market Research

There's two ways agencies can engage with a source: directly or indirectly.

Direct contact is where an agency is engaged in a verbal or written conversation with a source. This is problematic when that source *may be* a future competitor for your requirement. Sometimes this will generate more useful or substantive information, but is far riskier and more easily misinterpreted.

Indirect contact is where an agency reviews material without directly engaging a potential future competitor, or talks with a disinterested party. The majority of market research should be done indirectly. Not only is it easier by far, but it is less prone to bias from direct conversations, especially with potential competitors.

Requests for Information (RFIs) are a very popular market research method but consider their limitations before using them:

- Often RFIs are the only market research that an agency may conduct before undertaking a contract.
- RFIs usually consist of a set number of questions rather than a dynamic, evidence-based inquiry developed over a period of time.
- RFI responses are a lot of work for most businesses, but especially small businesses and/or the types of "innovative" businesses new to government that are sought after.
- Because of the way that RFIs are set up, the majority of information provided in any response is mostly marketing material that is reused for numerous RFIs, regardless of the agency or topic.
- Because they are labor-intensive and can often come at a considerable cost to the company preparing them, relying on them often increases the likelihood of a protest.

RFIs have a place in market research, but most of their value has been eclipsed by the advent of information available through the internet. It's possible to find more reliable information through indirect research on the internet.

Considerations for Sales and Capture Management

Whether agencies want to engage the market or not, chances are very high that if a government employee has anything to do with spending government money, the *market will engage them*.

Unlike many decades ago, there are now dedicated firms and resources designed to gather intelligence and package customer engagement solutions for companies to increase sales specifically for government agencies' budgets. Firms dedicate roles in departments for business development or "capture" in government with specific training regimens for effectiveness.

Salespeople can play an important role in helping to better educate potential customers about their company's offerings and helping potential customers reflect on their own needs.

But sales capture is also rife for potential abuse especially in the public sector where large amounts of government funds are being spent (the U.S. government is the largest buying entity in the world).

Government employees are working on behalf of someone else – the American public. This means that government employees have ethical and professional standards by which they must conduct themselves.

Numerous laws and regulations guide the communication or actions of government employees. If they don't adhere to these standards, such as FAR 9.5 for Organizational Conflict of Interest guidance, they may face civil or criminal penalties depending on the violation. This

could mean fines, suspension, firing, and, sometimes, even felony prison time.

However, rather than scare government employees off from conducting market research, these regulations should reassure government employees. They give government employees protections from the kinds of pressure that the sales process can engender in the average person.

Considerations for any direct conversations with contractors

- Any information that is shared in a conversation or a meeting could directly affect proposal preparation. By law, *all potential* offerors that could fulfill the agency's requirements must also have the exact same information, or they may suffer from an unfair competitive advantage. If information isn't shared properly, it could lead to a protest.

- All federal personnel have a responsibility to protect proprietary or confidential information, and not share such information with companies or potential competitors.

- Federal personnel must avoid appearance of commitment – only someone with a specific type of legal authority can obligate or commit the U.S. government through a contract (warranted Contracting Officer).

Sample approaches to use

Start every conversation with some variation on the following phrases:

> "Nothing discussed in this meeting authorizes you to work, start work, or otherwise obligates the government and is only for market research purposes. Any assumption on your part or on the part of your company is a mistake and has no effect on the government."

> "We are talking as part of ongoing market research / for market research purposes only. This conversation in no way obligates the government, or should make you believe that we have entered into a contract of any kind."

Fairness: Treat all potential bidders, vendors, manufacturers, and everyone else fairly and impartially.

Sunshine test: Imagine that everything done has a public audience. Consider whether an disinterested, casual observer would believe the government employee acted responsibly and reasonably.

Institutional knowledge: Reach out to and rely on colleagues and other experienced procurement professionals to learn best practices for conducting interactions, documenting any exchanges, and developing requirements for competitive solicitations.

Government employees must ensure that they keep all transactions with sales people professional, transparent, and courteous. But they also have the same responsibility.

Here are some common sales tactics agencies face:

Cold contact: Where someone you probably have never met calls or emails you about their company or offerings and how it will help you.

Name-dropping: Mentioning someone in a higher position or a supervisor, or referencing a conversation that may never have happened to try and gain influence with you or sway you in the direction they want.

Networked intro: Best way is to develop a good reputation with one customer and then make that customer work for you by introducing or referring them to other potential customers within an organization.

Big pitch: Much broader or organized effort to provide a well-thought-out presentation to get a bigger group of customers excited or interested in buying a solution.

Talking to potential competitors can be extremely valuable, as the Office of Management and Budget's memo, Myth-Busting: Addressing Misconceptions to Improve Communication with Industry during the Acquisition Process, points out.

Don't avoid these conversations. But instead go into them adequately prepared. Be able to recognize these conversations' pitfalls and persuasive elements.

Why do any of this?

Besides being a good idea to do extensive research before buying, the FAR legally requires[2] agencies to undertake some form of market research. If agencies have done an adequate amount of market research well, then they will be able to:

- Clearly communicate their needs or requirements to potential contractors
- Openly and transparently identify possible solutions or options
- Encourage innovation in designing and providing solutions

2 https://www.acquisition.gov/content/part-10-market-research

If agencies haven't done market research well, then they may end up:

- Being locked into inadequate solutions with poor quality contractors
- Inadequately describe requirements, that are then not met appropriately
- Hindering innovation
- Risking cost/schedule overruns
- Failing to meet mission goals

Depending on how extensive agencies want to search, and how much time agencies have to consider, research could take a considerable amount of time and effort. It's really at their discretion.

The size, scope, and complexity of the requirement will determine the amount of research and sources agencies gather information from. Should agencies take six months to conduct research for a product or service under the micro-purchase threshold? Probably not.

Should agencies spend a few months learning about custom software development projects, and find companies that seem to have a good track record before building a mission critical system? Absolutely.

Use the Agile contract format to procure Agile software development services

Waldo Jaquith, Randy Hart, Mark Hopson, and Vicki McFadden

CHALLENGE

Agencies are not structuring their contracts to support buying Agile software development services.

EXECUTIVE SUMMARY

- Legacy contract formats[3] that detail hundreds to thousands of software requirements up front are not suitable for Agile software development services.

- With the Agile contract template, available as a template, agencies should procure developers' time, as prioritized by the government product owner through an Agile cadence. Any contract must secure sufficient data rights for the agency in the work product or result of the development effort based on their mission needs.

RECOMMENDATION

For more than 20 years, using non-performance based formats, such as the Statement of Work, for professional services, has been strongly discouraged, as enshrined in FAR Part 37 - Service Contracting:

3 COs awarding contracts under FAR Part 15 must prepare solicitations and resulting contracts using the uniform contract format.

> (a) Performance-based acquisition (see subpart 37.6) is the preferred method for acquiring services (Public Law 106-398, section 821). When acquiring services, including those acquired under supply contracts or orders, agencies must –
>
> (1) Use performance-based acquisition methods to the maximum extent practicable

Performance-based service contracting (PBSC) emphasizes that all aspects of an acquisition must be structured around the *purpose of the work* to be performed with an objective assessment of contractor performance, instead of prescribing the manner in which the work is to be performed. It ensures that:

- contractors are given freedom to determine how to meet the government's performance objectives,
- appropriate performance quality levels are achieved, and
- payment is made only for services that meet these levels.

This proven methodology has yet to be fully implemented governmentwide for a variety of reasons, including inexperience in writing performance-based solicitations, cultural inertia, and concerns about more open and interactive communication with industry throughout the acquisition process.[4]

The problems caused by not using performance-based methods is especially acute when it comes to the software development professional services. Government solicitations to procure non-performance based custom software are often long and complicated, include many pages of requirements, and can take months — even years — to write.

4 https://obamawhitehouse.archives.gov/omb/procurement_guide_pbsc/

Structuring solicitations for Agile projects

A quicker, more outcome-oriented method, such as 18F's Agile contract format, would allow an agency to acquire an Agile software contractor with a solicitation that's only a dozen pages and written in an afternoon. It can be executed under existing procurement regulations and within Contracting Officers' (CO) existing authority. It can save enormous amounts of time, and the contract structure supports – not impedes – Agile development efforts.

The format is simple but contains a major shift in how agencies procure Agile software development services. In our experience, these elements are needed to align the solicitation with Agile software development best practices:

Use a Statement of Objectives — not a Statement of Work or Performance Work Statement[5] — because, with Agile, an agency doesn't know exactly what needs to be done, and can't possibly define it up front. The product owner, working with the development team, will determine on a sprint-by-sprint basis what work needs to be done. Anything else wouldn't be Agile.

Contract for time and materials, not a firm-fixed-price. In most cases when an agency is purchasing for a software project, they're not buying a defined product, but instead buying a software development teams' time. Agencies can use a time-and-materials contract, as the CO may prefer, or a labor hour contract.

5 According to FAR Part 37.601 a performance based solicitation may either be a performance work statement or a statement of objectives. In our experience, it is far more difficult to meaningfully pivot from years of non-performance based contracts, anchored with a Statement of Work, than it is to start anew with a Statement of Objectives. Very frequently, agencies will end up simply rename their former Statement of Work into a "Performance Work Statement" without making the fundamental changes necessary to the actual substance of their operations, administration, and partnering methods.

Historically, the default contract types for IT projects were firm-fixed-price, based on the assumption that this reduces risk and aligns with the way software licenses have historically been offered.

However, if an agency is in a position to constantly measure ("inspect and accept" in Agile) software quality, a time and materials contract — with a ceiling on total spending (or not-to-exceed ceiling) — allows for more flexibility for the software development team. A time and materials contract also allows for much easier escape clauses if the direction of the work changes or the contractor team does not produce quality software. If their work is inadequate, or their skills prove inappropriate, then no further work needs to be assigned to that contractor (effectively terminating the contract), and the contractor can be replaced. See our sample Determinations & Findings for more information on time and material contracts.

Have a short base period of performance, usually between 6–12 months. A couple of options to extend are fine, but keep them short, and never exceed a total contract length of three years. The agency is hiring a team to achieve a defined set of objectives and then leave. The government product owner should be heavily engaged throughout the project so contractor transitions should be relatively easy.

Maintain a nominal appendix of the backlog of user stories. User stories allow contractors to understand the specifics of the work that they're being asked to do, beyond whatever brief summary exists in the introduction. An agency must make it clear that the backlog has been included to illustrate the work that is presently believed will need to be done, but that it is *only* an illustration, and not a list of tasks that must be completed. It's a mistake to inventory user stories up front, rather than as part of performance. There's no way to know up front what work will need to be done, as is inherent to Agile (design, build, test; inspect; repeat). The language provided in the Agile contract

format allows for this flexibility, since the backlog accounts for evolving outcomes rather than a predefined list of needs.

Write a Quality assurance and surveillance plan (QASP). This is the cornerstone of performance-based service contracting. Rather than an exhaustive list, we focus on key, objective criteria that are able to determine and ensure quality.[6] For example, in our model, we require that at the end of each sprint, all code be complete, tested, accessible, deployed, documented, and secure.

Historically, the process of preparing software for use by its intended audience was complex, time-consuming, and risky. Standard industry practices have been automated making it simple, fast, and routine. Contractors must adhere to this practice ensuring agencies can deploy software themselves, without requiring highly technical staff to perform this trivial task. Allowing deployment to become complicated makes the agency dependent on that contractor, and makes it risky to replace them with a new contractor in the future.

We publish a sample QASP that can be incorporated as-is. We will continue to update this QASP as we discover additional, meaningful, and objective quality metrics.

Key personnel should include, at a minimum, the lead developer, and quite possibly the project manager. The agency is buying people's time, and wants to make sure that it's getting the people who were advertised. Be wary of specifying too many key personnel,

6 One of the most frequent misunderstandings is that the contractor should provide *their own* QASP. FAR 37.604 gives the government the discretion to do this if it so chooses, but it effectively means that the agency is ceding one of its most important ownership functions to ensure quality by allowing the party performing work to define their own measures of success. Think of it like a restaurant where they not only serve your food but also write your review of it to share with friends and family.

though — this requires that the contractors put those people on the project if they get the contract, which can mean that they're functionally benched until the contract is awarded.

In a truly Agile project, the project manager plays a very different role than in a waterfall project. Agile teams are self-organizing, which eliminates a major role traditionally played by a project manager. Instead, the project manager should serve as the contractor's interface to the client, be responsible for unblocking tasks and overseeing — but not controlling — the work performed by the vendor team. In many agencies, the project manager's role is usually required to prove that the agency is not engaged in personal services.

Allow for a distributed team, whenever possible. We advocate for remote development, rather than requiring the contractor team to be onsite at the agency's location. The best development resources are not in your city — they're spread out across your state and even the country. (There's a 150% difference in the salary of software developers between the most-expensive and least-expensive states in the United States.)

Using a modern technology suite, collaboration is easy. The product owner should always know what the team is working on, and the team should be readily available to answer questions or huddle on any issue that arises. They can do this with video calling for face-to-face collaboration (e.g. Google Meet, Zoom, etc.); instant messaging (e.g. Slack, Google Chat, etc.); task management (e.g. GitHub, Trello, etc.); and collaborative document editing (e.g. Google Docs, Office 365, etc.).

All work products must be published under an open source license — or, if using traditional proprietary code, the contract must secure sufficient data rights for the government in the work product to allow for use as intended and future development.

All work will be committed to a public code repository at least daily.

Keep technical proposals within page limitations, to minimize both contractor work and the time required for evaluation. Requests for more space may indicate that the contractor isn't taking an Agile approach, or that they don't adequately understand what they're proposing. In 18F acquisitions, we typically ask for responses to be 2-3 pages.

The contractor must submit links to 2–3 source code repositories where their illustrative past work can be seen. Allow this to include work done by key personnel (e.g., the technical lead) outside of their employment with the contractor, since many contractors will not have any public repositories to point to for lack of clients willing to work in the open. This is a far better indicator for how they are likely to perform under real-world conditions rather than the attempted simulation of coding challenges or hackathons.

Evaluation criteria should emphasize the contractor's proposed technical approach and their similar experience / past performance. An agency should hire a team based on their experience and their general approach to the work at hand. There's little else to go on other than these two criteria. This focus also keeps the contractor's performance work statement brief and to the point.

Contractors' key personnel must participate in a verbal interview process. This provides an opportunity to hear directly from the people who will be doing the work, rather than contractors' capture managers, and usually proves a quick way of separating the wheat from the chaff. This is not the type of conversation that is defined in the FAR as "discussions," or "clarifications," and does not allow for any revisions to the already submitted, written proposal. Instead it's a

critical quality control measure that confirms what was written down in the written proposal.

All of the above can be incorporated into a solicitation in a dozen or so pages, not including appendices like standard administrative clauses, a description of the technical environment that the contractor will be working within, the product backlog, etc.

You can put a solicitation like this together in a few hours. In a standard, three-day 18F procurement workshop, we use only the latter half of the third day to lead our client through an exercise in which we put together such a solicitation. With the right coaching, it's not hard.

Include the Contracting Officer (CO) in this process at every step of the way, starting as early as possible. When we host procurement workshops, we insist that the CO join the entire process, instead of just coming for the final day to hear about modular procurement and put together the solicitation. The CO needs to have the same base level of knowledge about the Agile development process as everybody else in the room. Without knowing about Agile software development, user-centered design, DevOps, etc., this prescribed contract format seems impossible.

Don't use hand-me-down government solicitations to procure Agile software development services. By focusing on procuring competent development team(s) to achieve clear objectives, your agency can produce solicitations in hours, not months.

Use time and material contract types for custom Agile software development services

Mark Hopson, Vicki McFadden, Alan Atlas, and Waldo Jaquith

CHALLENGE

Firm-fixed-price contracts are not appropriate for custom Agile development.

EXECUTIVE SUMMARY

- Use time and material (T&M) contracts with a not-to-exceed ceiling. If there is an engaged government product owner on the development team, they will know what the team is working on every day, so the historical risk of T&M contracts — costs spinning out of control — is reduced.
- Your team won't know the exact requirements for a software product before development, so don't use firm-fixed-price (FFP) for Agile software development.
- Don't measure Agile quality or outcomes by the completion of a number of sprints or user stories.

RECOMMENDATION

We recommend a time and material (T&M) contract for Agile software development with a not-to-exceed ceiling[7].

Flexibility is an absolute necessity for Agile software development. In an Agile project, there are no predetermined requirements listed as "shall" statements.

Instead, there is a product backlog comprised of user stories, which the product owner regularly adds to and re-orders to reflect each story's priority. Priorities can change at any time based on user research, changing agency needs, law or policy changes, etc. There is no way to know in advance what work will need to be done.

In Agile software development, an agency does not buy a defined product. Instead, it buys development teams' (developers, designers, content strategists, etc.) time. Therefore, the most appropriate contract type is T&M. The government, through an engaged product owner, can constantly prioritize the work to be done and measure software quality. The product owner should know what the development team is working on every day.

The traditional government fear of T&M contracts is that costs spin out of control. The empowered product owner[8] reduces this risk though daily communication, and frequent – every sprint – inspection of performance against a quality assurance surveillance plan (QASP). We also encourage using a not-to-exceed ceiling to put an additional cap on expenses to protect the government's interest.

7 FAR 16.601(d)(2) A time-and-materials contract or order may be used only if the contract or order includes a ceiling price that the contractor exceeds at its own risk.

8 https://18f.gsa.gov/2018/04/17/so-youre-a-product-owner/

In addition to flexibility, another T&M benefit is that it allows for development team flexibility to scale up and down as needed. Stable teams are a core tenet of agile development. Agile frowns on changing team structures more often than needed, but T&M allows the product owner to do so if necessary.

A real-life example: a new government product owner rightly determined that they could only manage one scrum team since the role was new. Six months into the project, the government product owner had mastered the role and was confident in their abilities. They decided to add an additional scrum team to speed development work. Once the key functionality has been built, the government product owner plans to return to one scrum team, likely smaller than the original. These changes are made possible by using T&M. No modifications are necessary because the agency is paying for the time humans spend building out the product – the number of humans and which roles are filled can change.

T&M allows agencies to save money, too. The contractor can only bill for time spent building a product. If there are delays hiring a new team member or a person goes on extended leave, the government is not paying for that time. T&M provides the opportunity for costs to come in under budget, which never happens with FFP.

A final T&M benefit is that if the contractor is not producing quality software – if their work is inadequate or their skills prove to not fit the work needed – no more work needs be assigned to the contractor, which effectively terminates the contract. There is no work in the product backlog, and the contractor cannot bill their time to the government. There is no need to terminate for convenience or cause.

To be clear, T&M does require more active management than the FFP "set it and forget it" contracts. The government must know what is going on, and review and approve invoices appropriately. Luckily, with

Agile development, this should already be happening. Read more about this shift in our recommendation on post-award management.

Find more information in our sample T&M determination and findings, which agencies can modify for their needs.

Labor hour contract

Why not a labor hour contract? Our experience reveals it's often necessary to buy additional materials for a software development project, such as cloud instances, software licenses, or SaaS fees. In those situations, it's better to pay the negligible costs for the contractor's materials; it will save a lot of headaches.

Firm-fixed-price contract

Don't procure Agile under a firm-fixed-price contract. We understand that most agencies discourage the use of T&M contracts in favor of FFP. However, Contracting Officers can use time and materials contracts 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. This is the case with Agile software development.

Agencies can FFP Agile. We just found it leads to a lot of headaches, fighting, and adverse outcomes. It is likely to go very poorly.

Firm-fixed-price contract types are perhaps the most commonly used methods to manage contractor performance. Government prefers the FFP contract type for a number of perceived benefits, and legislators and overseers encourage its use.

In this contract type, the contractor is rewarded when it delivers the exact list of requested features. An exact list of requested features becomes necessary up front, because using FFP contract types assumes that "performance uncertainties can be identified and reasonable estimates of their cost impact can be made, and the contractor is willing to accept a firm-fixed-price representing assumption of the risks involved."

When developing software using Agile development, the best requirements for a product cannot be known accurately before development, so FFP is not appropriate.

That being said, we've seen several programs *try* to FFP Agile, sometimes in elaborate and complex ways, and none of them have worked well. The most popular method is to FFP a sprint. Another commonly used method is FFP by story points.

Both scenarios result in unnecessary gamesmanship and fighting between the contractor and government. The contractor is incentivized to overbill and underdeliver. There are no canonical, objective ways to measure the number of user stories included in a given sprint or the size of a story point. So, when the government says "we think this story is size X" and the contractor says "we think this story is size Y" the resulting debate is difficult to resolve. These methods don't help to gauge contractor performance in any way. Agile quality or outcomes cannot be measured by the number of sprints or user stories completed.

An astounding number of projects are beleaguered by a desire to avoid using the most logical contract type (T&M), due to its reputation for cost overruns in projects that were not adequately monitored.

With FFP contracts, agencies don't have the flexibility to change the contractor software development team size over time based on product needs. If the contractor is not performing, agencies can't simply stop assigning them work and move onto another contractor — they must terminate for convenience or cause.

The only repeatable, standard, objective, scalable, universal way to measure work during a sprint is total effort-hours, which is not appropriate for firm-fixed-pricing. This *is* the purpose of time-and-materials or labor hour contracts.

Don't procure Agile under a firm-fixed-price contract.

Evaluate contractor proposals based on industry best practices

Waldo Jaquith, Randy Hart, Mark Hopson, Vicki McFadden, Kelsey Foley, Miatta Myers, and Stephanie Rivera

CHALLENGE

Traditional evaluation methods of custom technology practices are not based on industry standards.

EXECUTIVE SUMMARY

Review the strengths, weaknesses, and risks of contractors' proposals and then invite the most highly rated for a verbal interview.

RECOMMENDATION

Scenario

An agency has received contractors' proposals and their evaluation team is about to embark on the important mission of determining which contractor is best suited to meet the agency's needs.

This overview supports the evaluation team and helps them identify both indicators that show a strong proposal and red flags that indicate a weak proposal. This list is not exhaustive — every procurement is different.

Evaluation process

At 18F, we typically conduct targeted market research to identify a strong candidate pool of companies. As a best practice, we use the procedures allowed under FAR Part 8.405-2(c)(3)(iii)(B) to identify around ten potential companies interested in bidding before we issue any solicitation on GSA Schedule 70; this ensures that we solicit qualified and interested bidders. This method balances good competition and a reasonable number of proposals for the evaluation panel to assess.

Note: Your agency can only use this process if your agency intends to use GSA Schedules as their acquisition strategy. If your agency doesn't intend to use Schedules, your agency needs to follow the procedures required under the FAR that pertain to their acquisition strategy.

Don't use a point system to score proposals. Instead, each evaluation team member should review each proposal and list its advantages and disadvantages. Then, the whole evaluation team can discuss each proposal's advantages and disadvantages, and determine the strongest proposals.

A narrative review provides a detailed, defensible justification for the government's contractor selection in a way that numeric or color scoring schemes don't capture. It also allows the government to give feedback to the contractors that did not receive the award: the agency simply discusses the proposal's documented advantages and disadvantages. We encourage evaluation teams to use whatever materials they have available to make their decisions — websites, news articles, samples of prior work — and not only the proposals.

Invite the companies with the most highly rated proposals for an interview. This is when the agency can verify that the contractor can perform to the level proposed in their bid.

We ask that all named key personnel participate in these interviews. Each interview includes a timed, unstructured question-and-answer session, where the contractors answer questions about their proposal's technical aspects. Although we tailor each interview to the proposal, we often draw from an interview question bank that helps us plan interviews. This process allows the agency to better understand each contractor's proposed technical approach and to observe key personnel's interactions and working style.

Contractors will not be allowed to make presentations, ask questions, or change their submission in any way[9].

Determining the most highly rated proposals

When contracting for Agile development services, we typically recommend that agencies use four evaluation factors: technical approach, staffing approach, similar experience, and price.

The three technical evaluation factors, when combined, are significantly more important than the price. Evaluate the interviews as part of the technical approach, rather than as their own evaluation factor. For similar experience, ask the contractor to submit code repositories that are similar in size, scope, and complexity to the work that the agency is undertaking.

Below is what we suggest agencies consider when they're reviewing proposals using these evaluation criteria.

9 If these were permitted, the interviews could constitute a "discussion" under the Federal Acquisition Regulation, which could trigger an entirely different approach to the procurement process. These prohibitions are important.

Technical Approach

What to look for

- Competency. They should propose using the right tools in the right way and be able to defend their choices.

- A lack of novelty. The best approaches will recommend time-tested software and infrastructure, employing design patterns that are known to work.

- A lack of certainty. Maybe the contractor's idea is a good one, maybe it's a bad one — they can't really know yet, and they need to be aware that they can't know. A contractor will show they understand this by highlighting weak points or areas of uncertainty in their technical approach.

- A vision. The contractor needs to *see* the intended outcomes in a way that can act as a catalyst to the agency's vision.

- Understands program goals. The vendor should have a clear grasp of what the agency is doing. There should be no serious misunderstandings of information that was described clearly in the RFQ.

- Experience developing open source software.

- Collaboration and communication. The contractor expects the agency's product owner to be a valuable, active team member. They also expect to communicate proactively about risks and roadblocks, so they can work as effectively as possible.

- Regular and ongoing user research to understand users' goals and needs, and what to build that supports them. They will combine user research with usability testing to ensure that users can achieve their broader goals in using the software, and that it addresses their needs along the way. They plan to conduct user research, and test everything from rough prototypes to more polished software with actual users, throughout the entire design and development process.

- They follow a user-centered design process. They can explain how they make design decisions in relation to broader user goals and specific needs learned through their research.

- The contractor focuses on automation, reliability, testability, infrastructure as code, and other core DevOps principles. The proposal refers to modern automation and deployment tooling like Jenkins, Puppet, Chef, Travis CI, CircleCI, Kubernetes, Terraform, AWS, and Heroku.

Red flags

- Don't seem to understand program goals. They seriously misunderstand information that was described clearly in the solicitation.

- Misidentifying the name of technologies in such a way that shows a lack of experience communicating about them (e.g. "we'll index records with an Elasticsearch," instead of "with Elasticsearch," or "we recommend using JAVA," instead of "Java").

- Excessive complexity.

- They shirk page-limit rules (tiny fonts, reduced leading, etc.) because they believe their technical approach to be so brilliant that it can't possibly fit within the prescribed limit.

- Basing their solution on a fundamental misunderstanding of the agency's needs that they should have understood.

- Proposing the use of arcane platforms and technologies, especially when those arcane platforms and technologies are contractor specialties.
- They never once mention the software's accessibility, or do not identify how they will evaluate whether their software meets accessibility standards.
- They don't consider, explicitly or implicitly, that user research will ultimately determine the approach, which in turn will dictate the technical approach.
- Uses terminology like "requirements will be collected from the business owner"; user needs should be uncovered through research and listed as user stories in the product backlog.
- They're proposing to outsource what should be core competencies, e.g., DevOps or Javascript.
- They propose a process that includes working for long stretches of time without interacting with the agency and/or users.
- They describe the goal of research as being to "test the app with users," "find problems," or ask users what they "like," "want," or "might do" (shows that they draw conclusions based on what users *say* instead of observing and learning from users what they *do).*
- They use the term "user testing" instead of "usability testing" (not testing the user, testing the system's functionality).
- They propose relying on focus groups, instead of structured, one-on-one research interviews or usability testing sessions.

- They prioritize aesthetics over usability and usefulness, and cannot explain why they made design decisions.
- Don't mention anything about secure code practices.
- Don't demonstrate that testing is important.
- They propose long-term staff augmentation.

Staffing Approach

What to look for

- A small number of team members, each providing a clear value. Everyone proposed has a purpose.
- Familiar with and demonstrate use of modern software languages (e.g. Python, Ruby, PHP, C#, Javascript).
- Familiar with and demonstrate use of web-based application programming interfaces (APIs), especially REST and GraphQL.
- Experience using Git for software version control.
- The lead developer's skill mix and experience cover much of the work that the agency's project requires.
- If the developers have presences on social coding platforms (e.g., GitHub, GitLab, Bitbucket), how does their work look? What expertise is evident there? Do they have expertise that doesn't appear in their qualifications, but their work reveals?

- Staff qualifications support their claimed expertise. (For example, does the content strategist have any actual content strategy experience, or are they a project manager in sheep's clothing?)

- The lead user researcher's background indicates an understanding of how research can inform and shape strategy, design, and development; familiarity with a variety of research and testing methods; and experience deciding which method or methods to use based on the learning goals of the phase or needs of the project, and with recruiting users based on those goals and needs.

- The lead UX designer's background demonstrates strong craft skills and experience in generating concepts based in overall strategy, user research, and user-centered design best practices; and experience communicating those concepts visually via a variety of methods including but not limited to sketching, wireframing, prototypes, and more polished mockups to use in research and to guide development.

- Generally, the team is assigned to the project full-time and will not be splitting their time across other unrelated projects. There may be acceptable exceptions, such as for a scrum master or agile coach, but in general everyone should be fully staffed to the project. This is critical for developers, user researchers, designers, and all key personnel.

Red flags

- Overstaffing the bid. A team that consists of people with far more experience than necessary or more people than necessary means that the contractor either doesn't understand this way of working or is trying to over-staff the engagement.

- Proposing positions that do not belong in iterative development – business analysts, enterprise architects, delivery managers, etc.

- Poorly designed websites for the company, proposed subcontractor, or proposed staff qualifications.

- Proposing antiquated software technologies that don't have an active developer community (e.g. Cold Fusion, ASP, FoxPro).

- Lack of experience with test automation, aka DevOps, aka test-driven development (TDD).

- Insufficiently qualified lead developer.

- No apparent experience with usability research.

- No apparent experience with visual design.

- The flashiest team member is proposed to spend a tiny amount of time on the project.

- Key skills don't appear in any qualifications, such as:
  - Platform migration
  - Agile development practices
  - Automated (unit/integration/end-to-end) testing
  - Continuous Integration and Continuous Deployment
  - DevOps
  - Refactoring to minimize technical debt
  - Application Protocol Interface (API) development and documentation
  - Open-source software development
  - Cloud deployment
  - Product management and strategy

  - Usability research, such as (but not limited to) contextual inquiry, stakeholder interviews, and usability testing
  - User experience design
  - Sketching, wireframing, and/or prototyping, and user-task flow development
  - Visual design
  - Content design and copywriting
  - Building and testing public-facing sites and tools
- No actual technical staff, but "access to a database of resumes."
- Proposed staff don't currently work for the contractor, and there is no letter of intent from the proposed staff.
- Proposed staff qualifications are copied from the Internet, in large part or, more rarely, in whole.
- Key staff are not proposed to be 100% full time to the project, or the project is staffed with a number of partial FTE personnel.

Similar experience

18F often asks contractors to submit code repositories that demonstrate producing quality code that is in similar size, scope, and complexity to what the agency needs. If you do not have someone on your evaluation team that is familiar with code repositories, you should find one to serve as a technical advisor. 18F can support agencies with this need with a signed interagency agreement.

Technical evaluations should look for

- Proper use of Git, commit changes with personal accounts (not organizational), use of a branching / merging strategy, informative comments, evidence of code reviews, and use of a CI/CD pipeline.
- Code that conforms well to the solicitation's QASP.
- Git collaboration. Work was performed in a reasonable number of GitHub comments.
- Substantial projects. The projects were not created just to have something to point to for this RFQ.
- How they incorporate user feedback into their development process.
- Their tests are written well, and cover the supermajority of the code.
- Consistent, enforced code style.

Programmatic evaluations should look for

- Work that is conceptually similar to the agency's needs.
- Work that was centered on user needs, as opposed to leading with solutionism.
- Work that was completed by a team of a size that's similar to the size of the team that they're proposing.
- Design artifacts that show continuous and ongoing usability testing that indicate a user-centered approach to iterative design and development.

Red flags

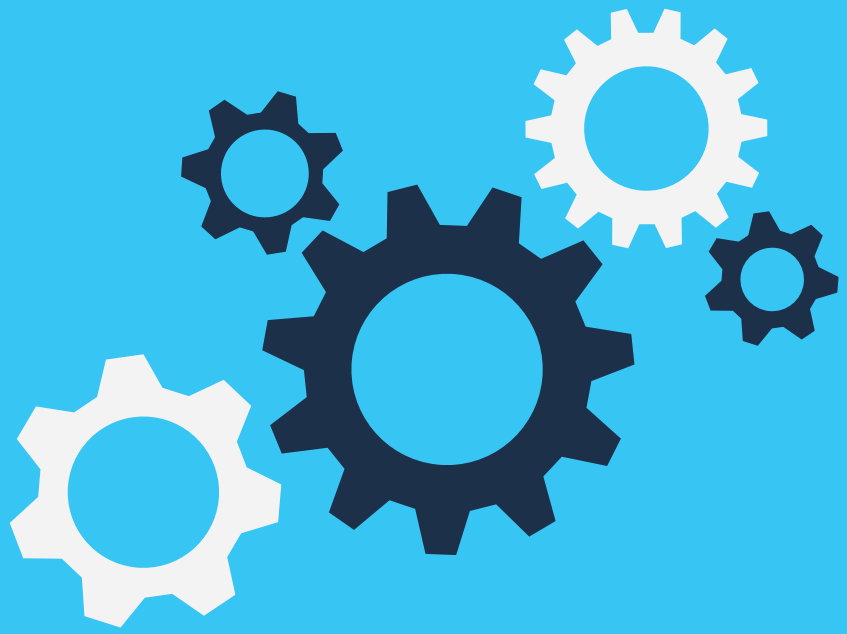
- The cited projects bear little evidence that the contractor created them.
- The projects are trivial.
- There's a finished product, but no code, or vice versa.
- The projects do not include good design artifacts and research plans.

This list is not meant to be exhaustive, but it gives the evaluation team some tips to help empower them to decide which proposals to evaluate as most highly rated and to bring forward to an interview.

Flexibility is an absolute necessity for Agile software development.

04 Doing the work

The following sections identify challenges and strategies to reduce risk during the post-award phase of custom technology projects in government.



Host an effective post-award kick-off meeting to energize folks for the work to come

Stephanie Rivera, Miatta Myers, Vicki McFadden, and Mark Hopson

CHALLENGE

Traditional post-award kick-off meetings usually do not adequately engage or prepare the contractor to begin work.

EXECUTIVE SUMMARY

A thoughtful post-award kick-off meeting should introduce new team members, empower the government product owner, plan next steps, and get the team excited about the upcoming work.

RECOMMENDATION

FAR Part 42.5 encourages Contracting Officers (COs) to hold a post-award kick-off meeting with the winning contractor. That is often treated somewhere between a check-box exercise and an afterthought.

We recommend being more intentional about the kick-offs, either as one combined meeting (contractual + project) or as two separate meetings scheduled in close proximity to one another. Some Contracting Officers have strong feelings about their procedures for a FAR-required kickoff, and prefer to hold their own separate meeting.

Goals of a post-award kick-off:

Make introductions: Start building a new team by getting everybody acquainted.

Share vision: Make sure everyone understands the work's purpose and impact.

Establish roles: Determine who will play what role on the agency and contractor teams; clarify the lanes for the Contracting Officer, Contracting Officer's Representative (COR), and product owner (PO); outline escalation paths.

Publicly empower the product owner: Make it clear to the team that the product owner is empowered to make decisions, that they are the agency's go-to person, and that the agency expects the product to evolve over time through learning and iteration with users.

Make time for the contractual kick-off: If this is held as a single meeting, cover whatever the CO would like to discuss — this can include security, legal concerns, and the COR's and the CO's contractual responsibilities.

Establish initial working practices: Discuss folks' preferred communication methods and working hours; plan tools and technology.

Plan next steps: Pick a few action-oriented tasks that get the team started on a positive cadence (e.g. scheduling Agile cadence, user research, technical prototype, etc.)

Get folks excited: Set the relationship off on the right foot, with the team leaving excited and energized.

To align both the government and the vendor, consider these things when planning a kick-off:

Attendance: Who will attend?

Leadership involvement: Does an agency leader want to kick off the meeting by saying a few words (e.g. underpinning the importance of this initiative, publicly empowering the PO, etc.)?

Stakeholder management: Who needs time on the agenda? Who needs to feel heard?

Agency preference:

- Does the agency have a policy for contract kick-offs?
- How does the CO/program office usually handle kick-offs?
- Would they like to have one meeting (combining the contractual kick-off and team kick-off) or two (separating these meetings)?
- What does the CO want to cover in the meeting (to prevent duplication)?
- How long are kick-offs, typically? Are they held in-person or virtually? If in-person, and the contractor is remote or distributed, how can these meetings accommodate remote contractors?

Contractor preference: Is there anything that the contractor would like to highlight/include in the agenda?

Facilitator: Who will facilitate the discussion?

Thinking through these goals and considerations should help make the kick-off productive and energizing.

Oversee Agile projects by measuring end user outcomes

Waldo Jaquith, Brandon Kirby, Vicki McFadden, Mark Hopson, and Alan Atlas

CHALLENGE

Current oversight practices can discourage Agile development.

EXECUTIVE SUMMARY

- Most oversight and governance methods for waterfall software development do not work for Agile software development. In fact, they make Agile development difficult.
- Requiring that project teams accomplish tasks by a specific date prevents them from responding to user needs and makes them build to predefined requirements.
- The only meaningful measure of success of an Agile project is the delivery of value to end users through working software.

RECOMMENDATION

Traditional models of project oversight are based on reaching specified milestones by specified dates. Oversight, in the government context, means determining whether those milestones are being reached on time and, if not, why.

Agile software development rejects this feature-deadline paradigm. Instead, Agile solves end users' problems by continuously researching and re-prioritizing planned work based on what that research uncovers.

For example, a project team may initially plan to complete a specific feature within a few months, but subsequent user research may reveal that the work is of low value, and not worth doing.

There is no way to marry the standard model of software development oversight with Agile software development — they are inherently at odds.

Even if it were possible, we would not want to keep this standard oversight model. Our research revealed that new Agile program managers spend most of their time not managing the product development effort, but working on reports to their oversight bodies and funders to document compliance. This documentation is likely to report things are going well, regardless of the actual project state, and offers little utility.

Agile software development means performing user research, documenting end user needs, performing work in the service of those users, and then presenting the work to assess whether it meets their needs. Observing the team's working practices and product's maturity over time is the best method of oversight.

How to oversee Agile software projects

The only meaningful measure of success of an Agile project is delivery of value to end users in the form of working software.

User value should grow constantly, every sprint. How a team measures that, whether through some type of quantitative metric or qualitatively through constant user feedback, will be team-dependent.

Sprint Reviews

Work quality is best assessed by attending sprint reviews to see the functioning software first-hand. At the end of every sprint, the team will demo the work performed.

Program or oversight staff can join sprint reviews periodically. Meetings are often held via video teleconference, so they are simple to attend. Seeing functioning software is much more useful than reading reports from other people who have seen the software. Initially, contract administrators may feel unqualified to evaluate the quality of the work performed, but they will gain confidence with experience.

The United Kingdom's National Audit Office's Governance for Agile delivery explains how to perform assessments, specifically section in 3.5, "Principles for governance":

> External assessment or reviews of Agile delivery should focus on the teams' behaviours and not just processes and documentation. Assessors are more effective in providing critical challenges if they have high-end skills, including technical and Agile delivery experience. In addition, they provide better value if they continually review how the team is performing, using observation as their main method of evidence collection. Key lines of enquiry for assessors include:
>
> the skills and experience of the team;
>
> the team dynamics – frequency and nature of communication inside and outside of the delivery team, and the level of input to the delivery team from the business;
>
> - the organisational culture – the level of commitment and openness;
> - the timing and nature of quality control by the delivery team – the testing and release framework;
> - the order in which the team tackled the tasks – prioritisation of actions and deliverables, the amount of actions in the backlog list;
> - the way the team changes its activity in response to the results achieved in each iteration; and
> - the value of outputs to the business.

Project Documentation

One of the Agile Manifesto's principles is "working software over comprehensive documentation," often expressed as "demos, not memos." Do not ask teams to produce documentation just for oversight and governance. This takes away from the time that they could spend delivering value to end users.

There are a number of documents that are likely to be produced and maintained continuously by the scrum teams that can be useful artifacts in the oversight process:

Product vision: A short description of the team's goal that they can use to quickly explain the product and galvanize support.

Product roadmap: This piece outlines the vision, priorities, and progress of a product over time.

Product backlog: This prioritized list of product features and bug fixes is usually written in the user story format.

Burn-down chart: After several sprints, the product team can start to project the amount of work remaining in the product backlog and estimate delivery time.

Burn-up chart: After several sprints, the product team can start to forecast approximately when certain functionality may be delivered or track completed work to total work.

Project risks: A list of conditions that could affect intended project outcomes; the team should actively work to reduce risks to minimize their impact on the project.

Data Rights Documentation

Documentation is also key to securing adequate data rights in the code that the team intends to retain, thereby helping to avoid vendor lock-in. While the contracting team will add the appropriate clauses to the contract, the responsibility falls to the product team to assign, or map, each deliverable or piece of code to the correct clause. This should be done on an ongoing basis as each piece of code or deliverable is submitted for acceptance.

For example, if the code is completely new – i.e., what the FAR calls "first produced in the performance of the contract" – its acceptance paperwork should state that it is delivered with "unlimited rights in accordance with the FAR clause at 52.227-14(c)(1)" (if that clause is in the contract). If the code or deliverable is derived from pre-existing software that has license terms attached – *including open-source software that comes with an open-source license* – the acceptance paperwork should state that the code is "delivered with the rights specified in the [associated license agreement], which constitutes a "collateral agreement" for purposes of the FAR clause at 52.227-14(c)(2)." Engage the contracting folks early and often to determine which data rights clause is in the contract and which part of it your code falls under.

Throwing in the towel

In six months, if a team has not delivered software with value to end users, meet with the project team to decide whether to continue the project.

The Office of Management and Budget writes, in their guidance for capital expenditures, that "[a]ll software development projects must produce usable functionality at intervals of no more than six months." If the team has not been able to deliver value to users in that timespan, it's time for serious reflection and possible termination for default[1].

Common antipatterns

Attempts to use traditional oversight processes with Agile software development tend to lead to the same common mistakes.

Do not do any of the following things.

- Forecasting when the software will be "done." The software will never be done.

- Having oversight bodies and stakeholders measure progress in ways that do not add value:

  Velocity: Over time, the product owner should keep an eye on this, but this measure is not useful for anyone outside the product team.

  Story points: Story points are a fictional currency that, again, may be useful for the product team, but not by anyone outside of the team.

  Lines of code written: These have nothing to do with user value in any way.

1 FAR 49.4 – https://www.acquisition.gov/content/subpart-494-termination-default.

- Pitting Agile teams against one another. There is no common metric to compare one team's progress and deliverables against another's. Don't try.

- Allowing scope creep.

- Forecasting the completion date of an epic or feature. The product owner should be able to say whether an epic/feature is something that will be addressed in the short-, mid-, or long-term. However, asking them to give the specific date when it will be delivered is asking them to lie.

- Watering down accountability. Always assign a product owner to an effort. Don't split responsibilities between a product owner, business analyst, project manager, governance board, etc.

- Having product backlogs in name only. Instead of the backlog detailing user value in user story format, many agencies have fake backlogs that are just tasks and requirements.

- Assigning an Agile methodology to a team, instead of letting them choose for themselves. Trust your team.

- Using proxies for user feedback. Strive for direct user feedback. Do not rely on the interpretation of "subject matter experts" or people who once did the job many years ago.

- Using monthly stoplight charts for cost, schedule, performance.

- Using a requirement traceability matrix. Asking for this is a big red flag for an Agile project. With Agile projects, the waterfall software development cycle concept of predefined "requirements" goes away and is substituted with "user stories" that describe the intended

outcomes of new features in terms of observed user needs. Requirement traceability matrices require teams to make formal change requests and rob them of their ability to rapidly react and adapt to changing needs, essentially undermining a user-centered approach to development.

On the other hand, user stories allow the team to prioritize the most important work to be done in the product backlog. They can also iteratively make changes to priority and what is in the backlog as they uncover user needs. This is expected in an Agile project, and no formal approvals are needed.

- Expecting teams to have a roll-out plan before the software is built.

No agency can move to Agile overnight. The shift requires a lot of difficult organizational changes. If those changes aren't made, a brave team might venture down the path toward Agile, only to be attacked and ultimately destroyed by organizational antibodies that have been trained to oversee and monitor a project with dated practices that have no place in Agile development.

Using the methods we recommend allows agencies to continue steering the ship while also allowing them to avoid the rocks.

See 18F's Agile principles and practices for more information.

Post-award contract administration looks different in Agile

Miatta Myers, Vicki McFadden, and Mark Hopson

CHALLENGE

If you want to use Agile (and Agile contracts), you need to be ready for the additional time and effort that working this way requires.

EXECUTIVE SUMMARY

Proper post-award contract administration requires more time and effort in Agile than in traditional government contracts. Working this way brings many benefits: more flexibility, more control of the product, more transparency into day-to-day work, sufficient data rights remain with the government so vendor lock-in is difficult, and outcomes are better – and delivered faster – for end users.

RECOMMENDATION

Status quo contract administration

Traditionally in government IT projects, awarding a contract is celebrated as a huge accomplishment. The winning contractor is expected to deliver all the requirements that the government has painstakingly detailed in the contract by the agreed-upon timeline and at the agreed-upon price.

The role of the government looks something like this:

Program or Project Manager: The program or project manager is marginally engaged, usually peeking in on performance through the contractor's monthly reports.

Contracting Officer's Representative: The Contracting Officer's Representative (COR) conveys messages between the program office and the contractor; usually these individuals manage many contracts at a given time and don't have the bandwidth to be intimately involved in a given contract's performance.

Contracting Officer: The Contracting Officer (CO) has most likely awarded a firm-fixed-price contract that is "set it and forget it," which makes approving invoices easy. COs across government are notoriously understaffed and overworked, so the CO will only get involved if there is a problem that can't be resolved by the COR or if a contract modification is required.

Contract modifications are common. A contract can go through dozens of modifications in its lifetime. These modifications can be both administrative ones like changing an official point of contact to a "within scope" change request.

There will be misalignments between what a program office intended as a contractual requirement and what the contractor interpreted the requirement to mean. During software development, the contractor or government will unearth new information that changes the intended design or functionality of the system. Laws and policies will change. Agency priorities and leadership will change. User needs will change. Technology will change.

The government and the contractor will need to negotiate all of these changes to the form and function of the system. The result will most likely come with a bigger price tag and an extension to the delivery date, especially with fixed price contracts. These static requirements and firm-fixed-price contracts cause government contract values to bloat and delivery dates to extend years past when they were originally imagined.

Unfortunately, this is standard on government software projects.

Agile contract administration

With Agile contracts, the contract award is just the first step that allows the real work to begin.

The role of government looks something like this:

Product Owner: The government product owner (PO) works closely with the contractor development team to iteratively identify, build, and deliver functionality. The contract period of performance is static. The functionality to be delivered is discovered throughout the project and varies. The price the government pays — because of time and material contract type — is variable, based on team size and hours worked, within the confines of a not-to-exceed cost ceiling. The government PO has discretion over when and what functionality is delivered to end users.

Transparency and open dialogue are paramount in Agile practices. The government PO is a member of the development team and plays an active role in setting the vision, prioritizing user stories, and clearing the team's blockers. Alignment between the government and contractor happens daily, so misalignments are quickly identified and resolved.

Contracting Officer's Representative: The COR may or may not be the government PO. The government PO knows what the development team is working on every day; if the COR is not the product owner, the COR knows what the development team is working on every sprint.

Contracting Officer: The CO is more involved than with traditional contracts; they are aware of contractor spending to approve invoices, and they check in with the COR to ensure that the contractor team is delivering value every sprint.

The contract scope is set at the product vision, not at the discrete requirements, so the team has flexibility to identify and implement user needs through the project. Aside from exercising an option, modifications are rare. The not-to-exceed ceiling likely will not change throughout the project. Value will be delivered to end users frequently throughout the project.

Agile contract administration requires more time and effort on the part of the government, both from the PO and CO.

But there are many benefits: more flexibility, more control of the product, more transparency into the day-to-day, less opportunity for vendor lock-in, and better outcomes for end users.

Monitor conformance with the QASP at the end of every sprint

Miatta Myers, Vicki McFadden, Waldo Jaquith, and Mark Hopson

CHALLENGE

The quality assurance surveillance plan (QASP) is different — and requires active oversight — in Agile development.

EXECUTIVE SUMMARY

- **Never** allow contractors to write the QASP.
- An Agile QASP ensures that code is tested, properly styled, secure, documented, deployed, and based on user research, at the end of *every* sprint.
- In addition to assigning a product owner (PO) to Agile development efforts, assign a technical lead to review code quality and conformance with the QASP at the end of every sprint. Ideally, this person is a government employee, but a contractor may do this work instead (as long as they're not on the same contract as the development team).

RECOMMENDATION

Status quo QASP monitoring

On most government IT projects, a quality assurance surveillance plan (QASP) specifies how the government will measure contractor performance/quality. Sometimes, the government allows the contractor to write the QASP for the contract. That is a terrible idea. It's like allowing a restaurant to write their own review. There are all sorts of ways for the contractor to manipulate the performance standards in their favor.

On most government IT projects, once the contract is awarded, government employees rarely look at the established QASP as a way to assess the contractor's performance. Performance against the QASP is usually a report that the contractor submits to the government monthly or quarterly. The only time that performance against a QASP is really closely watched is if it seems that the contractor is not performing; then the QASP is used as leverage to require a higher level of performance.

Agile QASP monitoring

The government should write the QASP and include it when it issues the solicitation.

Teams should monitor the QASP at the end of every sprint, and that the contractor be held to a high performance standard and quality level.

Specifically, we expect the code and documentation to be tested, properly styled, secure, documented, deployed, and based on user research. Teams can use the QASP that 18F uses on Agile development contracts and incorporate it as-is.

An Agile project's only meaningful measure of success is delivering value to end users through working software. Unfortunately, there is no quantitative way to monitor value to end-users each sprint. 18F's QASP is our best attempt to measure metrics that will impact the product outcomes.

In addition to having a government PO, we also like to assign a technical lead, ideally, a government employee. A contractor may perform this work as well, especially if there are no other options at the time, but they must be free from any conflict of interest. At the very least, this means that it cannot be someone from the same company, or even contract, as the Agile development team. The technical lead must be a neutral party.

At the end of the sprint, this technical lead will review all the code produced by the contractor team to ensure it conforms with the QASP before the code is accepted. If any of the performance standards are not met, the code will be returned for the contractor team to fix and resubmit.

This is not a full-time job, but will likely take 4–8 hours per scrum team, per sprint.

Software will never be done

05 Appendix

Verbal Interview Question Bank

Below are a set of sample questions to use in verbal interviews during acquisition to elicit additional clarity about a contractor's technical approach in their proposal.

Note: verbal interviews are not the same as oral presentations under FAR Part 15. Oral presentations allow a contractor to amend or change their proposal. Verbal interviews do not permit a contractor to amend or change their proposal; instead they provide an opportunity to ask clarifying questions on the technical approach portion of their proposal as written.

TECHNICAL APPROACH

Engineering

- Talk about your process for determining which software and programming language to use as your development team iteratively builds software products. We're also interested in hearing the rationale for the initial software and programming languages in your technical response.

- What is your technology stack of choice for this project, and why? Which technology stacks does this particular team have the most experience with? What other stacks/technologies are the team experienced with?

- Describe your technical development and collaboration process. Please specify your approach to version control, testing (and test-driven development), accessibility, continuous integration, and continuous deployment.

- Discuss the technical decisions you've made in your proposal, and what outstanding questions those decisions raised about this project. In particular, how do you plan to address the needs of this product's multiple user groups?

- Please discuss your approach to test-driven development and continuous integration and deployment.

- How will you approach technical oversight? How would you track the standards described in the QASP?

- How would you identify deep problems within a codebase? After identifying those issues, how would you address and reduce technical debt? What types of refactoring strategies would you consider?

- What do you anticipate as the largest risks in backend development?

- How do you intend to address data security needs and requirements?

- Tell us about a time you jumped in in the middle of a development effort. What challenges did you face? How did you overcome them? How do you envision integrating yourself within the existing development effort?

- Please describe your technical lead's experience with [programming language].

- Tell me about a system you built on top of some Infrastructure- or Platform-as-a-service.

- Tell me about an infrastructure problem that you helped solve (for example: slow application performance, unexpected downtime, a security breach; etc). What was the problem, and how did you solve it?

Collaboration

- How would you ensure good communication with all team partners?

- Walk us through how you see the designers and developers interacting as you build the product.

- Talk about how you'd like the [agency] to be involved as you design and build the product.

- What activities do you plan on engaging in to ensure a strong, collaborative process with regard to teamwork?

- Your Agile development approach looks to have a number of handoffs between different parts of the team. Can you tell us how you can ensure that the various members of the design/dev team will be able to stay on the same page during implementation?

- How do you typically communicate your findings and strategic recommendations to a client? How do you frame things when the findings might challenge some of your partners' assumptions?

- Have you ever worked with a remote/distributed team before?
  - **Yes:** What tools and/or mechanisms did you use to help promote open dialogue and foster communication between yourself and your teammates? How did you overcome any communication challenges and barriers?
  - **No:** What challenges do you anticipate? What do you think you'll need to succeed?

Staffing

- Talk about how you would quickly staff your design/dev team if awarded the contract.
- Tell us more about the different team roles you envision for this project.

Research & Design

- How have you incorporated changes based on user research?
- What do you see as the paramount user-centric feature(s) that users will interact with in the system?
- How would you design a usability test for an iteration of this product? What participants would you recruit? What tasks would you use? How would you analyze the results?
- How would you go about determining the visual feel and content tone of the project?
- How would you bring the full team and stakeholders into the research and synthesis process?

- How have you incorporated changes based on user research in other projects?
- Please discuss your experience with usability testing.
- Describe a time when you uncovered research results that disproved the team's core assumptions. What was the situation? What did you do? What did you learn?

Product & Strategy

- Talk about what you see as the three most important risks for this project and how you will help the [agency] to mitigate them.
- What do you need from the government product owner to make this project succeed?
- How will you develop a product vision and prioritize features? How do you envision ongoing feature prioritization working?
- Tell us about a project you led that was particularly challenging or complex. How did you approach it? How did it work out?
- What if an agency wanted something and you don't think they need it?

Iterative development

- Describe the Agile project management practices and tools you will use for estimating, planning, managing risk, team collaboration, and communicating status. Why would you use these?

- How will you keep developers, designers, and researchers engaged in building out a user story without lengthening the sprint or reverting to a “waterfall” development process?

- Talk through how you see the development team interacting with the [agency’s] product owner to ensure reasonably sized sprints for the development team.

- How would you alert the government product owner if the team encounters tasks that require more work than originally anticipated that can’t be completed in the current sprint?

- Tell me about your experience with Agile or other iterative development styles. How does practicing Agile affect the technical choices you make?

Based on the answers, also dig in on anything that is pertinent on a given project, or that seems odd in a given proposal, for example:

- Tell us more about what you envision will happen during the month-long system definition kickoff.

- Talk through your rationale for why you think PHP will be the right fit for what [agency] will need to build.

- Explain your choice of datastore technology in your technical approach. Detail both benefits and problems.

- What do you see as the main challenges of building and testing an API, and what steps do you recommend to overcome those challenges?

- What are some of the compromises you think will need to be made in meeting performance budgets in mobile devices? How do you intend to address those compromises? What does mobile responsiveness mean to your team?

- What is your risk management strategy regarding PII?

- How do you intend to secure data-in-transit using a FIPS 140-2 compliant encryption module?

- How will your technical approach account for low bandwidth environments with limited or old technology?

- What was your thinking behind choosing labor categories that did not directly correspond to the labor requirements?

- Can you please describe your experience with Pay.gov?

NOTES

The only meaningful measure of success of an Agile project is the delivery of value to end users through working software