The Agile Ecosystem:
A Guide to Maximizing your Investments in Agile Transformation

Agile Six Applications, Inc.
Brian Derfer, CTO
Dan Levenson, Digital Services
Prologue

Agile Six is extremely optimistic about the recent developments in emerging digital service delivery practices at all levels of government. A growing Civic Tech movement has enticed contributions from a diverse group of brilliant people, with a common goal of building a better civic experience for constituents. For example, the U.S. Office of Federal Procurement Policy (OFPP) recently announced the launch of a Digital IT Acquisition Professional (DITAP) Program. Following two successful pilot classes, this program equips Contracting Officers to buy digital services, delivered in a more iterative, modular and human-centered way. The OFPP also advised agencies that by the year 2022 any technology purchase greater than $7 million must be handled by a contracting officer or specialist that has earned this FAC-C-DS certification. This policy will be an important mechanism in incentivizing agencies to adopt a more agile and human centered approach to procurement as part of the broader ecosystem required to facilitate a successful Agile adoption.

Introduction

Many city, state, and federal agencies are pivoting towards Agile development practices in order to improve the quality, shorten the time to market and reduce the cost of digital services. We believe that a successful transition towards Agile is best addressed as an evolution towards a new ecosystem rather than merely an adoption of a specific set of practices or body of knowledge. An overly narrow focus on Agile ceremonies and practices, without a plan for addressing how these practices interact with and are supported by other external elements in the overall development ecosystem, prevent organizations from maximizing their investment in Agile and place the successful adoption of Agile at risk.

In this paper, we attempt to map out some of the important elements of this broader ecosystem, introduce a lexicon for describing the interaction of these elements with each other and with Agile teams, and provide guidance on adopting Agile as an ecosystem.

Background

Agile Six is involved with transitions towards Agile in both commercial and government organizations. In our experience, there is a tendency, especially in the early phases of a transition, to focus too narrowly on Agile practices and ceremonies without accounting for how they interact with other elements of the development ecosystem. Agile training practices and literature support this tendency.
Consider: An aspiring scrum master or product owner can get a certification entirely based on memorizing Agile concepts and ceremonies as if these existed in a theoretical bubble, without having to demonstrate the practical application of these in an actual team or organization, and without understanding how these practices intersect other areas of the IT ecosystem, such as acquisition, culture, or governance. A licensing body would never grant someone a pilot’s license based purely on someone’s performance in a flight simulator, but that’s exactly what we do with certified scrum masters and product owners.

Figure 1: Agile practiced in a theoretical bubble, distinct from the other aspects of the ecosystem.
Not coincidentally, when an Agile project encounters difficulty, a common response is to look inward at an Agile team’s practices for answers – Are our sprints the right length? Do we have the right retrospective format? Is our team too large? While these are legitimate questions, it is critical to also explore the larger ecosystem within which the project is situated. Does the team operate in an organizational culture that supports Agile? How has the project been structured and provisioned? Does the system’s architecture support frequent releases? What governance structures constrain how software is released?

The larger context often determines both how Agile principles and practices are received by practitioners as well as the extent of the effectiveness of Agile practices on organizations and projects. It is useful to think of Agile practices as operating within an ecosystem that contains elements either synergistic or antagonistic to Agile principles, and it is often the larger ecosystem, not merely the practice of Agile ceremonies in isolation, that determines whether Agile will thrive in an organization. Just as a species’ survival depends on its compatibility with the ecosystem – think of polar bears in San Diego – Agile practices can thrive or struggle depending on the ecosystem in which they are practiced.
The Agile Ecosystem

Figure 2 represents several important elements of a typical software development ecosystem, as well as some attributes of those elements that tend to be synergistic with or antagonistic to Agile practices.
Let’s walk through each element of the Agile Ecosystem:

**Culture:** Organizational and team cultures that promote a high degree of individual autonomy, local decision-making, emotional trust, and collaboration are more synergistic with Agile than cultures that are highly directive, control-focused, and overly reliant on titles and silos of responsibility.

There are a few of reasons for this. First, Agile frameworks rely on the communication of intent and high-level acceptance criteria via user stories instead of detailed requirements specifications. This mode of engagement relies on thoughtful team members “filling in” the “missing” pieces of information based on a creative synthesis of several pieces of information – the Product Owner’s overall vision for the project, an empathetic understanding of the users’ and other stakeholders’ needs, and of course the acceptance criteria specified in the user stories. In the desired end state – not to mention the essential nature of coding, testing, or designing – work on Agile teams is almost always heuristic and creative, rather than rote or algorithmic. It requires local autonomy for team members to make decisions real-time based on multiple pieces of information. “Theory Y” organizational cultures that promote a high degree of individual autonomy and reinforce

---

**Shu Ha Ri**

Developing an Agile mindset requires individuals and organizations to embark on a journey through various stages of learning and mastery, which Japanese martial arts refers to as the stages of *shu*, *ha*, and *ri*. During the *shu* phase, practitioners “learn by doing” – faithfully following the formulas and recipes developed by others in order to acquire knowledge. The next phase, *ha*, occurs when practitioners have internalized the formulas and begin to modify them to better suit their specific problems or context. Lastly, in the *ri* stage, practitioners have not only internalized and modified the forms, but also have mastered a mature and nuanced understanding of their underlying principles as well. It is at the *ri* stage that practitioners, unhindered by structures of formulae or templates, can innovate freely, creatively and expertly adapt, and most fully realize the value of their practice.

Early adopters of Agile, including many federal agencies, often see the *shu* stage – the ceremonies, roles, and tools – as the beginning and end of their Agile transformation. But without an understanding of the principles underlying Agile – the nature and value of human interactions, the value of empirical feedback loops, and the nature of complex tasks – organizations may devolve into a control-based distortion of Agile that hinders the delivery of positive outcomes.

Beware of *shu* and *ha* Agile coaches. The shiny certifications may be enticing, but coaches who lack a nuanced understanding of Agile often have an adverse impact and can demoralize an Agile transformation. The improved outcomes a *ri* coach can deliver more than justify paying a premium.

* Sutherland, Jeff and Sutherland, JJ. *The Art of Doing Twice the Work in Half the Time* (New York: Crown Business, 2014)
emotional trust create conditions far more supportive of the kinds of collaboration and local decision-making that Agile requires than “Theory X” organizational cultures.

Second, Agile frameworks rely on self-organizing teams, which in turn rely on the establishment and maintenance of emotional trust among team members. Rather than a project manager holding team members accountable for meeting commitments, delivering quality, and creating the conditions for success, the team members hold each other accountable. As Patrick Lencioni describes in “The Five Dysfunctions of a Team,” people are better able to hold each other accountable when the team has established a high degree of emotional trust.

Finally, at Agile Six we subscribe to the idea that people and teams perform optimally when they are challenged with purposeful work, given a high degree of ownership and local control over how that work is accomplished, and are well-supported in a trust-based environment. There are various lines of research and literature that support this idea, starting with Maslow’s hierarchy of needs and reinforced in various ways by scores of other authors, including Dave Logan, Dan Pink, and Edward Hallowell. This blog post provides a summary of this literature and the consequences it has for creating optimal organizational cultures.

In our experience, most civil servants come to work for their government filled with purpose, but many lose it somewhere along the journey of their careers. Promoting the wrong people based on seniority or technical skill rather than leadership ability, coupled with rigidity in the personnel system, from hiring and retention practices to control-based performance evaluations that limit autonomy, can incentivize behaviors that are antagonistic to Agile. Fear of failure, often manifested in short-sighted, risk-averse behavior, ends up replacing purpose within most government systems. Without a willingness to embrace failure, there are harsh limitations on the ability to learn or innovate. The result is an ecosystem that induces a workforce that is limited in its ability to deliver real value to citizens. To truly change outcomes, there must also be a transformation in the fundamental way governments operate at an organizational and cultural level.

**Access to Users and Stakeholders:** Agile prioritizes people and interactions over tools and processes. An organization and process structure that emphasizes collaboration, minimizes work silos, encourages transparency, and facilitates a high degree of access and communication among stakeholders, users, and teams will be more synergistic with Agile than an organization and process structure that emphasizes titles and clearly circumscribed areas of responsibility, relies on prescribed contractual relationships to mediate most or all interactions, and attempts to control and constrain communication among stakeholders, users, and teams.
While this point might seem fairly obvious in theory, it is surprising how infrequently it is put into practice, especially on government contracts, where tightly controlled and constrained access and communication are the norm. For example, we were recently working on a project where all communication between contractors and government personnel – even for simple clarification on requirements or questions regarding the intent of a user story – required initial written communication via a database, followed by a formal scheduled meeting attended by far more people than were needed to answer the question. Such long and formal feedback loops discourage the kind of free-flow of ideas and information between stakeholders that facilitate success on Agile projects. They also reinforce a very formal relationship between contractors and agency personnel, which can feel almost adversarial at times. Conversely, we are currently working on a different project with a small federal agency where continual and ad hoc communication among stakeholders is the norm. The informal and open communication we have established on this project has been extraordinary for a government project and continues to pay both tangible and intangible dividends.

**Acquisition:** An acquisition strategy that is based on clear objectives and measurable outcomes, can support and incentivize incremental delivery, and is team-centric rather than project-centric is more synergistic with Agile than a process that specifies in detail how work is to be accomplished, is structured around large deliverables packaged in a single release, and/or forces the continual assembly, disassembly, and reassembly of teams around disjointed chunks of work.
Let’s take each of these elements in turn. First, one of the reasons Agile works well on heuristic tasks is because it better supports local decisions based on an understanding of intent supported by empirical feedback. In other words, if a team understands the vision of the project and the intent of a user story, it is in a far better position to implement that vision and intent – based on a detailed understanding of “facts on the ground” – than anyone would have been at the start of the project. For example, a feature requirement that is specified in detail eighteen months prior to its implementation may become outdated for any number of reasons: it could not have predicted technical innovation, it could not have predicted what user feedback regarding its value had surfaced, it could not have predicted changes in the underlying business case for its implementation, etc. As a result, it is generally much more beneficial for an Agile team to clearly understand the high-level intent and objectives of a project than it is to have a detailed plan or set of instructions provided at the outset.

Second, for the same reasons, the larger the project is – both in terms of the length of time that is required for delivery and in terms of its scope – the more difficult it is to respond to ongoing change. Smaller projects, delivered on relatively smaller timelines, are in a far better position to leverage ongoing technology evolution, respond to changing business needs, and maximize the value of empirical feedback loops in Agile.

Finally, much of the value of Agile is realized by people working as teams to collaborate on solutions. As a result, the most effective teams are those that have succeeded in creating a collaborative, trust-based team culture. Creating healthy teams takes time and investment; it is not unusual for it to take several months to establish the level of emotional trust on a team to make it truly effective at self-organizing. The time it takes to create a healthy team can be at odds with a project provisioning strategy that prioritizes small projects and short releases, as the team could finally be getting into a healthy rhythm just as their project is ending.

Agile Acquisition vs. Acquisition for Agile

“Agile Acquisition” is an increasingly used buzz phrase in the federal procurement space. At Agile Six, we make a clear differentiation between Agile Acquisition and merely the acquisition of Agile services. Many agencies are transitioning to acquiring Agile services but have failed to address or understand the broader Agile ecosystem that supports these services and creates the conditions for favorable outcomes.

Agile Acquisition, at its best, is concerned with supporting an Agile ecosystem, including Agile values like collaboration and transparency in how we procure goods and services, as well as iterative feedback loops that support the measurement of project outcomes in terms of the value they deliver to users, and continual adaptation to improve these outcomes. Unfortunately, the existing federal procurement system is too often focused on minimizing risk, avoiding failure, and measuring work rather than outcomes. Not surprisingly, this has led to a high number of failed Government IT projects, barely usable products, and reduced trust of government by citizens.
One answer might be to focus on provisioning teams and feeding them with a steady stream of work, instead of on provisioning projects and feeding these to different teams. Our partner Robert Rasmussen has written a white paper that provides the broad outlines of how this might work.

**Architecture:** In general, architectures that are modular, loosely-coupled and service- or API-centric tend to be more synergistic with Agile than architectures that are monolithic, tightly-coupled, and application-centric.

The reason for favoring the former has to do with what we call the “blast radius of change” to a system. Any change to a large, tightly-coupled monolithic web application is likely to result in a large “blast radius,” with reverberations both predictable and unpredictable to other areas of the application. Both the size of the application and the nature of the coupling determine the size of the blast radius. The larger the blast radius, the greater the need for automated test coverage, more thorough regression testing, and/or a higher tolerance for bugs. Put more simply, the larger the “blast radius” of changes within an application or system, the larger the testing footprint must be. At some point, a tightly-coupled monolithic architecture may impact a team’s ability to do frequent code releases with any confidence or efficiency. Less frequent releases may, in turn, lengthen feedback loops and reduce the team’s ability to be responsive to users, or adapt to the market. In this way a tightly-coupled monolithic architecture can be thought of as antagonistic to Agile.

In a more modular architecture – comprised of numerous smaller components or services, each of which can, at least in theory, be deployed independently – each change results in a smaller blast radius. As a result, there is less need for large regression testing efforts with each release, and less chance for bugs to appear in far-off areas of the application. Moreover, the efficiencies gained by a smaller blast radius allow code to be released more frequently with greater ease, which helps teams realize other benefits of Agile, such as responsiveness to users’ feedback or the ability to leverage ongoing technology evolution within the application. For this reason,
modular, loosely-coupled architectures, both within and between applications and services, tend to be synergistic with Agile.

What about the interdependencies among services or applications? When a module or service changes, won’t other areas of the system that depend on it need to change (and hence be deployed and tested) as well? It depends. Changes to the interface of a service or module – i.e. changes involving the manner, structure or nature of information being exchanged – will necessitate accommodating changes to other components. But code changes or fixes that do not entail changes to the interface will not necessitate accommodating changes in other areas of the application. In effect, the interfaces and APIs provide boundaries that constrain the blast-radius of change.

One final area of service and application architecture deserves mention: Dependency Inversion is a principle of software design that supports loose-coupling of components by removing the normal dependency that higher-level components have on lower-level components. Interactions and dependencies among components are mediated via abstract interfaces instead of direct references to concrete implementations. This design has two benefits: First, different implementations can be provided to an application provided they fulfill the “contract” defined by the interface. In this way, dependency inversion constrains the blast radius of change – limiting the blast radius to whatever changes are made within the component supporting the interface. A second benefit is that dependency inversion supports mocking, which in turn supports automated unit testing. As we’ll see below, the extent to which processes, including testing, can be automated is an important element in an Agile ecosystem.

**Operations and Infrastructure:** IT infrastructures that support the automation of key development processes, on-demand resource provisioning, and continuous integration tend to be more synergistic with Agile than infrastructures that are highly manual, subject to tightly-controlled resource provisioning, and where application integration is intermittent and/or infrequent.
Many of the benefits of Agile frameworks are realized via short empirical feedback loops. Sprint reviews, retrospectives, and daily standup meetings, for example, can all be viewed as Scrum ceremonies that provide a structure for surfacing and/or responding to feedback regarding the team’s activities. In this context, other practices and tools that shorten feedback loops align well with Agile. It’s not surprising that practices and concepts such as DevOps and SecDevOps have arisen along with Agile frameworks to support these short feedback loops.

**Governance:** Governance processes that are lean, accessible to the team, and are engaged early and continuously throughout the development of a system are more synergistic with Agile than governance processes that are heavy, remote, and are only engaged late in the development lifecycle.

Consider a cumbersome release approval process, involving multiple independent approval workflows and remote stakeholders who have been engaged only minimally in the details of the project, if at all. No matter how efficient the team is at developing high-quality code, their development efforts essentially run into a brick wall when it comes time to release to production. Previously unengaged stakeholders may not only take considerable time to get up-to-speed on the project, but they may also impose new technical requirements that require changes to the code. In sum, an antagonistic governance process can thwart a team’s ability to practice Agile effectively.

In contrast, a governance process that contains fewer independent approval bodies and workflows, which includes stakeholders who have been continually engaged with the project, and which is leaner and more efficient, is likely to facilitate frequent releases, which in turn may provide richer engagement with users and a development process that can respond more easily to users’ needs.

**Automated Toolchains Shorten Feedback Loops**

Over the last couple of years we have been working with a toward a more Agile-friendly ecosystem. As part of that effort, we have adopted a CI/CD infrastructure that includes GitHub as a version control repository, CircleCI as a continuous integration and task management engine, and AWS Elastic Beanstalk as a cloud deployment and provisioning service. Code that is checked into our GitHub repository automatically initiates a build, which includes the automatic execution of all unit tests within the project, as well as deployment to a “QA” environment. This automation allows the team to obtain feedback regarding the quality of the application, including any bugs or compilation issues, very quickly, which in our experience leads to increased quality as well as reduced time spent fixing bugs. This rapid feedback loop, when combined with a modular service-oriented architecture, supports the ability to iteratively adapt software very quickly – that is, true agility.
Understanding the Ecosystem at Work

Consider the following scenarios:

- A Program Manager on a government software development contract is trying to implement iterative and incremental development with frequent releases, but must navigate a software governance process wherein each release must be approved by fifteen different governance bodies. It takes at least two months for each release to wind its way through the approval process.

- A Product Owner is working with a Scrum team who commits to a scope of work every two weeks during the Sprint Planning meeting, yet is working on a fixed scope contract with a specific delivery date. She can clearly see that the team’s current velocity is not sufficient to complete the entire scope of deliverables by the delivery date.

- A Scrum team working on a large web application finds that it is frequently releasing regression bugs to production that do not seem to be directly related to the code worked on during the sprint. The team is frustrated because it does not seem humanly possible to perform regression testing over the entire application each sprint.

- A Contracting Officer has awarded a fixed-scope Statement of Work (SOW) based contract for agile development services and is regularly encountering scope issues due to the iterative prioritization of work based on user feedback. This leads to numerous, time consuming contract modifications that hinder the development team’s ability to incorporate meaningful feedback, resulting in a poor user experience, increased spending on useless features, and an increased time to market.

These and similar scenarios are quite commonplace, and it would be surprising if readers who have worked on Agile projects have not encountered at least one of them. When such problems occur, there is often a tendency see Agile development practices as the root of the problem, and hence to look for answers within Agile itself:

- The team could deploy releases internally each sprint and gather several internal releases into a production-bound release every few months, then push that release through the lengthy approval process.

- The Product Owner could create a roadmap that lays out the remaining deliverables for the entire project and review it with the team, and perhaps ask for the team’s input on how to achieve all the deliverables on the roadmap.¹

¹ Such meetings occasionally yield interesting insights or strategies, but usually end up as a decision to sacrifice quality or a request to the team to work more hours.
• Perhaps the team could move to 4-week or 6-week sprints, which would enable them to perform the full set of regression tests within each sprint.
• The Contracting Officer might attempt to modify the SOW to be more predictive and stipulate user stories are “locked in” long before completion.

We refer to such solutions as “Agile Adapters,” since they bridge a gap between an Agile practice and one or more elements of an ecosystem that are antagonistic to Agile. Agile Adapters are discussed in more detail below.

Another potential reaction is to perceive that Agile has been tried and “just doesn’t work for us.” This reaction is especially common in organizations that have been practicing waterfall for a long time and are undergoing a nascent transformation to Agile. When Agile practices are inserted into an ecosystem that is not supportive of Agile, and then these practices don’t work as expected, it is a natural tendency to conclude that Agile practices are at fault – not that the benefits of Agile can only be realized if other elements of the ecosystem are adjusted as well.

Both Agile Adapters and the outright rejection of Agile practices may result from a failure to consider the larger ecosystem within which Agile is being practiced. Conversely, understanding the success or failure of these practices as conditioned on the relationship between the practices and the ecosystem, and not simply the result of something inherent within the practices themselves, may result in better questions, more thorough answers, and Agile implementations whose value is more fully realized. It may not be that something is inherently wrong with the polar bear; perhaps the San Diego desert simply does not provide what the polar bear needs to thrive.

To be clear, the point is not to suggest that certain elements of an ecosystem cannot coexist with Agile. Of course, large, tightly-coupled monolithic applications can be developed and maintained using Scrum, for example. The relationship is subtler, effectively captured by notions of synergy and antagonism. An element of an ecosystem that is synergistic with Agile may help a team or organization more fully realize the value of investments made in other areas of the ecosystem. For example, the value of investments made in automated testing may be more fully realized when combined with a modular architecture that supports dependency injection. Similarly, the value of access to users or stakeholders is likely maximized in an organizational or team culture that places a high value on trust, empathy, and collaboration. Conversely, multiple antagonistic elements can combine to create real problems. For example, a large monolithic application that lacks both automated testing and a CI infrastructure could present challenges to frequent releases, which in turn could undermine the ability of a project to respond to change.
In other words, synergistic or antagonistic elements of an ecosystem can combine to produce interesting cancelation or multiplier effects. For example, an unwieldy architecture, combined with a testing, integration and deployment infrastructure that is not automated, combined with a cumbersome release approval process could render effective iterative development nearly impossible. The inability to push code regularly could partially negate the value of other synergistic elements of the system, such as a high degree of access to users. Conversely, an organizational culture that promotes autonomy and collaboration could be rendered nearly inert by an ecosystem in which access to users and stakeholders is limited. Finally, a contracting process that provisions a team and feeds it with work in a way that facilitates continuity and cohesiveness, combined with an organizational culture that promotes collaboration, combined with a lean collaborative governance process has the potential to support a thriving Agile practice.

**Agile Adapters**

Agile Adapters are practices that are introduced into a software development ecosystem that bridge the disconnect between antagonistic elements of the ecosystem and Agile practices. The following are some examples of Agile adapters that we’ve encountered:

- **Increasing sprint lengths** to adapt to an ecosystem that does not support frequent releases well (often because of antagonistic elements within the architecture or infrastructure, or because of limited access to users and stakeholders).
- **Bundled releases** to production that include multiple sprints worth of work rather than releases each sprint (often because of an antagonistic governance process and/or antagonistic elements within the architecture or infrastructure).
- Performing **user-testing with “stand-ins”** rather than actual users (often because access to real users is constrained, or because of an inability to efficiently deploy code to an environment where user testing can occur).
- Requests to the team to **work more hours** or intentionally **incur technical debt** to meet a deadline (usually because the project has been provisioned as a fixed scope-fixed price contract and there was a mistake made in predicting the scope and/or required resources).
- Utilization of **“testing sprints”** or **“bug fixing sprints”** that are focused entirely on shoring up existing code rather than delivering features (often introduced because quality is not being adequately addressed each sprint, perhaps due to a lack of an effective automated testing infrastructure, or because of a decision to incur technical debt).
• **Retroactive adjustments to story point estimations** at the end of a sprint to better align them to the work which was completed (often done to satisfy the management and reporting requirements in an organization with a directive management culture).

• Awarding *rigid and prescriptive contracts*, often for long performance periods and large dollar values, to large system integrators for Agile development services (often due to the risk-averse culture prevalent in contracting, compounded by a lack of Agile training and subsequent understanding of Agile values and outcome-based thinking of the contracting staff).

To be clear, the point is not that Agile adapters are bad. In fact, they are often necessary for an Agile project to survive in the face of antagonistic elements of an ecosystem. However, Agile adapters may provide a kind of “process smell” that indicates that there are strong antagonistic elements elsewhere in the ecosystem. Identifying the adapters and understanding their causes may provide strong clues towards making improvements in the ecosystem that will allow teams to maximize the effectiveness of Agile.

**Transforming Your Ecosystem**

Clearly, antagonistic elements can exist within many different layers and areas of a process or organization, often far away from an Agile team’s immediate sphere of influence. As a result, continuous improvement practices such as sprint retrospectives – despite the many benefits they provide to a team – may be insufficient to address all of the risks, challenges, and frustrations that may confront a team in their attempts to deliver a project using Agile, a point we at Agile Six also make in our white paper outlining an Agile Risk Management Framework.

Successful adoption of Agile is a multi-dimensional endeavor, covering a number of different areas and practices within an organization. As we alluded to above, there are some practices that can help to create and sustain an ecosystem that is supportive of Agile:

• Take a “human-centered” approach to Agile transformation by understanding, in an empathetic and non-judgmental way, the organization’s current ecosystem and capacity for change. Organizations are value systems that in many cases have accumulated their values over decades. It is not realistic to think that an organization that has used a control-focused management approach (perhaps successfully in many ways) will be able to suddenly internalize an entirely new value system based on trust and autonomy. Agile transformation entails playing the “long game,” and in general, the larger an organization is and the longer it has been around, the more “inertia” there will be
around its existing value system. In our experience at Agile Six, understanding the current organization and taking an empathetic approach are essential first steps to encouraging change.

- Invest in understanding why Agile works, including the human psychological and motivational factors underlying Agile, such as trust and autonomy. Understanding why Agile works, along with understanding the organization’s current value system, may manifest some fundamental cultural and motivational divergences that will need to be acknowledged upfront. In our experience, there are often conscious and unconscious attempts to undermine Agile by stakeholders in the current value system. Understanding where these are coming from and why they are occurring can prompt healthy conversations. Conversely, not understanding and acknowledging them can slow or sabotage an Agile transformation. Agile Six’s training curriculum for Agile and Scaled Agile starts with the human motivational factors underlying Agile and weaves this theme throughout the entire training curriculum. This curriculum has been highly successful at CMS and is a great starting point for an Agile transformation.

- Adopt procurement and provisioning practices that focus on building flexible, scalable contracts that incentivize the right behaviors and allow the flexibility to incorporate validated learning throughout performance, in alignment with the other facets of the Agile ecosystem.
  - Flexible: Use Statement of Objectives (SOO) to align with an agile approach to delivery. The use of outcome-focused objectives allows the backlog to be reprioritized based on feedback and validated learning, ensuring continuous value is delivered.
  - Scalable: We recommend that most contracts use a team-based capacity model that consists of a base Contract Line Item Number (CLIN) for each performance period along with optional “surge” CLINs that allow for additional capacity, often an agile team of six to nine members, to be exercised at the Government’s discretion. Additional capacity may be required due to user feedback, validated learning during performance, or new legislation or policy.
  - Incentivize the right behaviors: It is important to clearly identify the program outcomes and align the acquisition strategy accordingly. What type of behaviors do you want to incentivize? Use flexible, scalable contracts and task orders that don’t get in the way of program delivery and incentivize behaviors like accountability and innovation. Where Cost-Plus contracts incentivize a staffing approach, often using low to moderately skilled personnel, and offer little accountability toward meeting outcomes, an FFP type contract offers the vendor flexibility to provide high-skilled and talent dense personnel and adjust the labor mix during performance to ensure contract objectives are being met and value is
being delivered. Additionally, an FFP type contract incentivizes innovations, like automation, that may lead to better quality and more rapid delivery, as the potential reduction in staff does not harm vendor bottom lines as it would under a Cost-Plus contract.

- Encourage open communication and access, and remove fault-lines and barriers between people on the same team or project who happen to work for different organizations. Adopt a “one team” mentality, regardless of organizational affiliation.
- Emphasize that user research and human-centered design are crucial components of Agile Acquisition. Acquisition teams should conduct collaborative and transparent market and user research to focus on designing, developing, and awarding human-centered contracts that account for the broader agile ecosystem. In our experience, most acquisition teams lack a product vision or connection to users. When working with federal acquisition and program teams, we begin with a visioning workshop that stresses the concept of bringing the users into the room, which may be Grandma at CMS, or the Veteran at the VA. Focusing on users as humans not only leads to the creation of better solicitations and subsequent contracts, but also unlocks the purpose in the acquisition team by cultivating a meaningful understanding of the critical importance of their daily work.
- Encourage architectures and patterns that support modularity, loose-coupling, and dependency inversion, and adopt an API-first approach to development.
- Continually identify opportunities for shortening empirical feedback loops. Process automation and the adoption of a DevOps culture and mindset can be valuable tools in creating short, efficient feedback loops.
- Understand that a risk-aversion (fear of protest) too often drives the acquisition approach, not fully realizing the fallacy in putting significant effort into attempting to control something that can never be completely controlled. While sound decision-making and clear and concise documentation are important, vendors will often protest a contract award as a function of their business model, regardless of how the procurement is managed. Protests also occur due to a lack of transparency and collaboration from the Government acquisition team, which may cause vendors to believe nefarious activities are occurring. Conducting a transparent and collaborative procurement, from the identification of a need through contract award and delivery, has proven to be an effective way to minimize protests.
- Adopt a lean governance mindset and process by reducing the number of different governance processes and bodies involved in a project or organization. Identify all stakeholders and their interests as early as possible in the project lifecycle, and encourage early and continual participation by all stakeholders.
• Align the entire organization behind Agile. The fact that Agile ceremonies are focused mostly on the development team should not absolve other elements of the organization from understanding Agile principles and practices, and committing to supporting them. In our experience, one of the strongest predictors of success in transforming an organization towards Agile is commitment to Agile principles by organizational leaders (both informal and informal) and achieving buy-in throughout the organization. Not every member of the organization needs to be a certified Agile practitioner, but some basic training and understanding of the principles and rationale underlying Agile goes a long way. Because there can be tremendous inertia resisting the adoption and internalization of Agile principles, especially in large organizations such as federal government agencies, this is by no means an easy transformation to achieve. One method, which a number of federal agencies seem to be following already via “Digital Service” initiatives, is to carve out specific areas of the agency – including personnel, process, infrastructure and governance – that can support an Agile ecosystem and put firewalls between it and other areas of the agency that may not be ready to support an Agile ecosystem. The idea is to incubate a small thriving Agile ecosystem within a large organization, then gradually expand the boundaries of that ecosystem, rather than trying to transform the entire ecosystem at once.

• Facilitate regular meetings or other forms of engagement that provide an opportunity for stakeholders supporting different areas of an organization, or even different organizations, to discuss challenges and possible solutions. Sprint reviews, for example, should be public ceremonies in which stakeholders from throughout the organization participate and provide input into a project. Unfortunately, these potentially valuable sources of collaboration are too often attended solely by team members. Other project-centric meetings that invite communication and collaboration across different stakeholders, such as the Program Increment Planning meeting in SAFe, can prompt discussions and illuminate potential synergistic and/or antagonistic elements in an Agile ecosystem. Finally, Agile Communities of Practice are a form of engagement that can provide a regular forum for discussing challenges and introducing perspectives from a wider audience. Agile Six has been deeply involved in the creation of the Digital Services Coalition, a community of practice and advocacy group comprised of a number of federal contracting companies who are committed to supporting Digital Transformation, including the creation and support of Agile ecosystems. Increased engagement among companies, and between industry and government vendors who share a common set of values around Agile transformation, might provide a counter weight to some of the inertia in government resisting Agile transformation described above.

• Begin identifying and tracking Agile Adapters in your organization and teams with an eye towards eventually eliminating them (or more specifically, eliminating the underlying
antagonistic elements of the ecosystem that are causing them to be required in the first place). Recall that Agile Adapters are mechanisms to cope with antagonistic elements of an ecosystem, so identifying them and discussing why they are needed can often lead to the identification of more fundamental issues. For example, one of the most common Agile Adapters is to increase sprint lengths due to the recognition by the team that they cannot get to a satisfactory “definition of done” within the sprint timebox – i.e. their code may not be well-tested or deployed at the end of the sprint. Rather than assuming that the time to test and deploy code are fixed variables, it might be helpful to investigate why these tasks are taking so long. For instance, it could be that testing time can be shortened through automated testing, which in turn might lead to productive discussions about whether the system is architected to support automated testing. Or it could be that the scope of regression testing can be reduced by moving towards a more modular architecture and thus reducing the blast-radius of a given set of changes. The team might also consider moving to a more automated build and deployment infrastructure – which again might lead to other discussions about whether architectural or infrastructural elements of the ecosystem are synergistic or antagonistic to Agile.

Conclusion

Companies and organizations in both the commercial and government sectors continue to adopt Agile frameworks, principles and practices. During the last few years, federal, state, and local governments have begun transformations towards Agile along with other modern industry best-practices in software development. To maximize the benefits of a shift to Agile, and in some cases to ensure successful adoption of Agile to any degree, this industry-wide transformation would benefit from understanding the ecosystems within which Agile projects operate.

This paper has outlined a basic framework and lexicon for thinking about how a software development ecosystem supports or fails to support Agile practices. It presents a few examples of the kinds of elements that exist in this ecosystem, as well as some ways in which those elements can be either synergistic or antagonistic to Agile and can combine in complex ways to impact the effectiveness of Agile practices. It suggests that certain common practices surrounding Agile projects are actually “adapters” that help to facilitate Agile practices in the context of antagonistic elements of the ecosystem. Their presence may be a signal that something in the ecosystem needs to change to better accommodate Agile practices. Finally, we’ve proposed some specific steps that organizations can take to create a thriving Agile ecosystem.
A successful Agile transformation requires the courage to not let perfection be the enemy of good. It necessitates embracing vulnerability, a willingness to fail (and learn from failure), and a degree of transparency that is often at odds with the traditional approach to delivering Government services, which is often opaque, risk-averse, and confusing. Embracing and nurturing an Agile ecosystem will empower citizens and taxpayers through a more user and human-centric Government that better understands the solutions we are creating to support them.

**Need Help?**

Agile Six works with both government agencies and contractors to provide training and transformation support to thrive in this ecosystem. Through our work with several agencies and civic tech firms, we have developed an extensive curriculum and coaching services based on the topics covered in this paper. If your agency or company needs to transform your ecosystem, or equip your staff with the knowledge and tools to understand Agile within these new ecosystems, please reach out to us.

We offer the following services to both civic agencies and commercial companies:

- Full stack software development services
- Agile Scrum Training & Coaching (including certifications)
- Scaled Agile Training and Coaching (including certifications)
- Digital Service Acquisition Training
- Procurement and Acquisition consulting for federal, state and city buyers
- Proposal and Solutioning consultation for firms retooling to the space
- Organizational transformation for executives, leaders and teams
- Agile Governance and Product Management
About the Authors

Brian Derfer is the CTO of Agile Six Applications, whose mission is to transform the working relationship between government and the private sector — to empower more human-centered and value-driven collaboration, to deliver better services and digital experiences, to improve the value of IT expenditures, and to create a more responsive government. Brian blends nearly 20 years of hands-on experience in the software industry with a background in cognitive anthropology in order to bring a unique naturalistic and human-centric perspective to technical problem domains. Prior to joining Agile Six, he was involved with several start-up ventures as founding partner or early associate in vertical markets that include healthcare, education, and social media. His main areas of focus include Agile frameworks, software development best-practices, domain-driven design, artificial intelligence, and web and mobile development. Brian holds a Master’s Degree in Cognitive Anthropology from UC San Diego, and is a Certified Scrum Master through the Scrum Alliance. He is named on three US patents. His other interests include soccer, sailing, music, history, and human cognition.

He can be contacted at brian.derfer@agile6.com

Dan Levenson is the Senior Vice President of Digital Services at Agile Six. Dan is a respected leader in the Government Digital Service Transformation space, with a laser focus on making incremental progress towards positive outcomes. Dan was formerly the Digital Service Contracting Adviser and a Contracting Officer at the Centers for Medicare and Medicaid Services (CMS) and held a Digital Service Specialization in Contracting. As the CMS Digital Service Contracting Adviser, Dan championed human-centered Agile transformation throughout the Agency, acting as a liaison to other departments and to the vendor community, and introduced innovative processes, structures, and procurement approaches to reduce impediments to Agile deliveries. As a Contracting Officer he was passionate about building flexible, scalable contracts that incentivize the right behaviors, while practicing Agile values, like collaboration and transparency, throughout the acquisition process. Dan was named to the 2017 and 2018 FedHealthIT100 as one of the most influential individuals in driving change and advancement in the Federal Health Information Technology Market. He is recognized as “one of the pioneers in this new Agile way of contracting by breaking down the barriers of the traditional way of procuring IT services.” Dan holds a B.S. in Logistics and Supply Chain Management from the Robert H. Smith School of Business at the University of Maryland, College Park. In his free time, Dan enjoys hiking, sports, spirited political discussions and spending time with his dog Finn.

He can be reached at dan.levenson@agile6.com