Agile Advocate: Top 3 Impediments to Agile

What's stopping CIOs and CTOs from realizing the potential of agile and DevOps?

Bill Drew
Thu, 10/05/2017 - 18:39

The Agile Advocate is a series of thoughts or musings on the agile movement and DevOps in particular. It was motivated by the seminal work of social philosopher Eric Hoffer, "The True Believer" — at least in structure. It's not meant to be a manifesto, but simply a series of thoughts and reflections coming from over three decades of developing and delivering software and systems of a wide variety to an even wider variety of users and customers. My aim is not to evangelize or convert, but to provoke and stimulate discussion.

Ask any federal chief information officer or chief technology officer about their reasons for adopting an agile posture in their development and delivery of capabilities to their end users. They’re likely to cite the importance of adopting agile
and how it’s serving as a fundamental change agent in their culture. They’ll talk about how it changes the client relationship and improves their ability to satisfy their customer’s needs. They’ll discuss the focus on service delivery and the collaboration among and between teams to meet the end user’s needs and how adopting an agile posture has allowed their organization to gain and maintain a real sense of relevancy with the end-user community.

But off the record, they’re often more likely to cite some real impediments to realizing the potential of agile and DevOps adoption.

I’m going to present here what I consider, at least at this point in time, the top 3 impediments to the CIO/CTO preventing organizations from realizing the full potential of agile processes and the adoption of DevOps practice and principles.

**Compartmentalized Contracting**

So, we’ve embraced the DevOps principles and practices. We made those hard organization changes and structured and organized our teams of contractors based on services. These cross-functional teams are singularly focused and engaged on delivering that service or capability to their end user from development to operations — “we build it, we own it.”

And as these teams progress, as one would expect, they realize there are opportunities to collaborate and share functionality and capabilities outside the team, to leverage and benefit from work done by other teams and contractors. And this is good thing. After all, sharing and collaboration, swarming to a solution as a team and project is a DevOps principle.

But this collaboration involves more than simply reaching across team or service boundaries, but also across contract boundaries. And herein lies the challenge.

For while we preach collaboration and sharing within and across teams as a common practice and tenant of DevOps, we contract on a siloed, compartmentalized basis — task oriented within the requirements scope of the contract. If we need to reach out and collaborate across teams, there’s no way for other teams to charge their time or account for time and resources spent in those endeavors.
At the end of the day, government contractors have to satisfy the terms of the contract; answer the “musts” and “shall” as agreed upon with the government. But with the current contracting structure, there’s really no way to accomplish that and still meet their contractual obligations.

**Process Over Product**

Agile methodologies and frameworks have introduced a structured, defined and generally well-understood approach to the software development process. The intent of these agile methodologies is to support the production of software, help teams deliver working software incrementally and maintain relevancy with the end user. And as in all things, when applied appropriately, it has proved to be a significant contributor for organizations looking to bring more and relevant functionality to their users.

But just as we do with many things, if a little is good, then more must be better. And with that thought, the agile process and their “ceremonies” have tended to grow unabated within a project.

First, the daily standup starts to expand, from 15 minutes to 30 and beyond. Multiday planning sessions begin to resemble extravaganzas rather than well-organized, streamlined opportunities for collaboration and swift decision-making. Then meetings about meetings seem necessary and reporting becomes more important than actually doing the work.

And as the production of software begins to lag under the weight of the process, the gut reaction is to add more meetings and processes and agile resources. So, we add more resources: coaches, mentors, facilitators and layers.

The process and maintaining the information to report on the process begins to overcome its original intent — to streamline the delivery of software or the product. The end result is we find ourselves devoting more effort to recording, reporting on and discussing what should be done and what we’ve tried to accomplish than actually getting something accomplished and delivered.

At this point, the process has become the product.
Lack of Continuous Integration

In short, CI provides the repeatable, predictable processes and supporting components and infrastructure that allows for the continuous integration of code and the building or construction of your application.

That sounds like a simple enough high-level description that’s hard to argue about concerning its value. But what does your CI actually do? What does your CI produce? And how does your CI support the agile posture and increased cadence I’m looking to establish?

I wrote in a previous article about the importance of understanding what your CI actually does for you. As the article states, you’ll find that in most projects that profess to have CI, it is undefined, completely lacking requirements and largely left up to the technical team to determine scope and capabilities.

This lack of requirements often results in a gap in functionality, implementation and ultimately expectations. And the most critical function of the CI, the integration and integration testing are most likely to be left out. Without this, product owners and testers are unable to continually interrogate and validate the application and provide that critical continuous feedback to the development teams.

And without this continuous feedback loop, your CI is reduced to waterfall.

Road Forward

Looking at these impediments, none of them are insurmountable. They’re all solvable, especially with the talent and resources in the federal space.

When it comes to contracting, there’s plenty of creative and resourceful contracting professionals who can craft the necessary provisions for an “agile” contract to support cross-contract collaboration. But first, they need to be alerted up front that it’s an issue in order to address it. Once they are, I’m confident there will be a common provision established in federal contracting to support cross team collaboration.

When it comes to the use, or the overuse of agile, this needs to be viewed in the same way we do software refactoring. We need to continually assess how our agile
ceremonies are addressing the primary reason for their existence — to facilitate the
development and delivery of relevant functionality to our users. We need to refactor
the application of agile methodologies and ceremonies in the same way we do our
software, continually assessing and repositioning them to better achieve their goal
— the delivery of capabilities to the end user.

Concerning CI, we simply need to do the difficult work of determining our
requirements up front. We cannot, as I’ve pointed in a previous article, leave the
implementation of such a critical component up to the assumptions of a third party.
Do the work, spec out the requirements as you would any software deliverable and
verify the implementation meets your expectations.

These may be the top impediments at this point for me, but there are several more
that could have made the top 3. I will be looking at some of these in upcoming
articles and assessing them in the same way we did here. Stay tuned.

This is the fifth column in a series. Read the others here:

Paving the Way to Do DevOps

Leveraging the Cloud

What is Infrastructure as Code and Why Do I Care?

How to Involve Operations in a DevOps World

-- Sign up for our weekly newsletter to receive the latest analysis and insights on
emerging federal technologies and IT modernization.

View printer friendly version

agile

The Agile Advocate