July 10, 2022

What’s Happening in Agile?

By Christopher Stahl

I recently had a chance to catch up with Mark Schwartz, the CIO at U.S. Citizenship and Immigration Services (USCIS) in the Department of Homeland Security. Schwartz is one of the leaders in the government's move toward agile software development, whom I first met when he was a participant in a Kennedy School Executive Education Program.

My goal in following up was to find out from him whether there had been any important developments in agile software development in the government, in the last year or two. He didn't disappoint.

Schwartz referred to a major development called "Continuous Delivery" (one aspect of a somewhat larger concept called DevOps). The basic idea is to shorten the cycle-time for new software releases from the several weeks of an agile sprint (which, itself, is obviously a huge improvement on traditional, government processes) to one day – or even, at some point, several times a day. This is done, by making the requirement that developers are working on at a given time truly bite-sized -- a single discrete change in the system -- and by automating the delivery process.

Having the changes come in these tiny chunks has advantages over traditional agile development, not to speak of older, waterfall processes. Any time some new capability is fielded, there is a risk it won't work correctly. The larger and more complicated the release, the greater the risk of a problem, resulting in the need for costly and time-consuming rework (something that has always bedeviled waterfall-type development). If very small capabilities are added each time, very quick feedback from users is possible, and fixing problems is far easier, because there is less sunk cost in what has been fielded. This approach also avoids the problem that arises with multi-week, agile sprints that deliver multiple features. Even with agile, real bottlenecks can occur as some new features are ready earlier, but need to be held off to batch into what comes out of a sprint.

Continuous Development got its big boost in 2009, when representatives from Flickr, the photo-sharing service, now owned by Yahoo, spoke at a conference about doing ten, new software releases a day. With the speed obsession of leading tech companies, there has developed a sort of "Can you top that?" competition -- Amazon apparently can push out thousands of releases a day.

The government is unlikely ever to become Amazon (indeed, Amazon's tech competitors have trouble becoming Amazon), but certainly by government standards, Continuous Delivery displays a lot of agility. At this point, in the government, only USCIS (and some work supported by 18F) has projects that are delivered every day. Yet Continuous Delivery looks poised to spread soon elsewhere in government, especially given a recent hire by 18F of one of the leading experts on the topic in the tech world.

To make continuous delivery possible, many aspects of the process of getting the new code fielded must be automated, such as testing, deployment, configuration, and security. That's what USCIS has done. Oversight of software also needs to be done in a new way for Continuous Delivery to work. From the overseer's perspective, Continuous Delivery has advantages, because the tiny releases are lower risk and the ability to see immediately whether a new feature is working increases transparency.

USCIS has put in place a policy encouraging frequent deployments and laying out the appropriate control processes for them. For example, a team that uses Continuous Delivery must be able to roll back its deployments quickly, if anything goes wrong and must manage all its code, test scripts, and deployment scripts in the agency's version control system.

Schwartz says that at this point, all his new projects are using Continuous Delivery.

And in an amazing coup for the government, 18F has just hired Jez Humble away from his position as Vice President of Development Velocity for Chef, a Seattle-based automation company for IT infrastructure and applications development. Humble is the author of the book ***Continuous Delivery*** and a lecturer at the UC Berkeley School of Information.

Schwartz believes the pace of recruitment from the tech world into 18F, USDS, and agency-based digital services is accelerating, something for which he gives former, federal CTO Todd Park, who now lives in the Silicon Valley, much of the credit. Getting people from the tech industry into government can develop a snowball momentum of its own, as each new recruit makes it easier for other techies to make the leap.

These people are being mostly brought in as term-hired Schedule A employees, with few becoming civil service lifers. We may be on the cusp of an important transition in the federal career path hiring model, the development of a short-term (but non-political) career track to complement the traditional civil service "stay in government all your life" model.

I have advocated for this idea for a long time, given the changes in values among young people who expect to have several different careers during their working years. It is exciting to see this new model beginning to emerge. While it is appropriate for lots of government jobs, it is particularly so in the tech area to compensate, at least partly, for the government's difficulty, for salary and other reasons, in recruiting young, tech talent.

What is Continuous Delivery?

Continuous Delivery is the ability to get changes of all types—including new features, configuration changes, bug fixes, and experiments—into production, or into the hands of users, safely and quickly in a sustainable way.

The goal is to make deployments (whether of a large-scale distributed system, a complex production environment, an embedded system, or an app) predictable, routine affairs that can be performed on demand.

This is achieved by ensuring that one's code is always in a deployable state, even in the face of teams of thousands of developers making changes daily, eliminating the integration, testing, and hardening phases that traditionally follow "dev complete," as well as code freezes.

Why Continuous Delivery?

It is often assumed that if one wants to deploy software more frequently, one must accept lower levels of stability and reliability in one's systems. In fact, peer-reviewed research shows that this is not the case; high-performance teams consistently deliver services faster and more reliably than their low-performing competition. This is true even in highly regulated domains, such as financial services and government. Continuous Delivery provides an incredible, competitive advantage for organizations that are willing to invest the effort to pursue it.

The practices at the heart of Continuous Delivery help achieve several important benefits:

Low-risk releases. The primary goal of Continuous Delivery is to make software deployments painless, low-risk events that can be performed at any time, on-demand. By applying patterns such as blue-green deployments, it is relatively straightforward to achieve zero-downtime deployments that are undetectable to users.

Faster time to market. It's not uncommon for the integration and test/fix phase of the traditional phased software delivery lifecycle to consume weeks or even months. When teams work together to automate the build and deployment, environment provisioning and regression testing processes, developers can incorporate integration and regression testing into their daily work and completely remove these phases. This also avoids the large amounts of re-work that plague the phased approach.

Higher quality. When developers have automated tools that discover regressions within minutes, teams are freed to focus their efforts on user research and higher-level testing activities, such as exploratory testing, usability testing and performance and security testing. By building a deployment pipeline, these activities can be performed continuously throughout the delivery process, ensuring quality is built into products and services from the beginning.

Lower costs. Any successful software product or service will evolve significantly over the course of its lifetime. By investing in build, test, deployment and environment automation, substantial reductions in the cost of making and delivering incremental changes to software are achieved by eliminating many of the fixed costs associated with the release process.

Better products. Continuous delivery makes it economical to work in small batches. This means developers can get feedback from users throughout the delivery lifecycle, based upon working software. Techniques such as A/B testing enable developers to take a hypothesis-driven approach to product development, whereby ideas can be tested with users, before building out whole features. This avoids the 2/3 of features normally built that deliver zero or negative value to a business.

Happier teams. Peer-reviewed research has shown Continuous Delivery makes releases less painful and reduces team burnout. Furthermore, when releases are more frequent, software delivery teams can engage more actively with users, learn which ideas work and which don't and see first-hand the outcomes of the work they have performed. By removing the low-value, painful activities associated with software delivery, focus can be placed upon what is cared about most—continuously delighting users.

If this sounds too good to be true, bear in mind: Continuous Delivery is not magic. It's about continuous, daily improvement—the constant discipline of pursuing higher performance by following the heuristic, "if it hurts, do it more often, and bring the pain forward."

To Learn More, Contact Us Today

Christopher Stahl

CEO

Email: chris.stahl@stahlcompanies.com

176 Wardensville Grade

Winchester, VA 22602

(240) 701-2434 Direct Number

Web Sites: www.stahlcompanies.com

www.stahlusa.com

Charles Shillingburg

Director, Business Development

Email: charles.shillingburg@stahl-companies.com

176 Wardensville Grade

Winchester, VA 22602

(703) 728-9059 Direct Number

Web Site: www.stahlcompanies.com

www.stahlusa.com