Architecting for **Agile DevOps**

OrbusSoftware

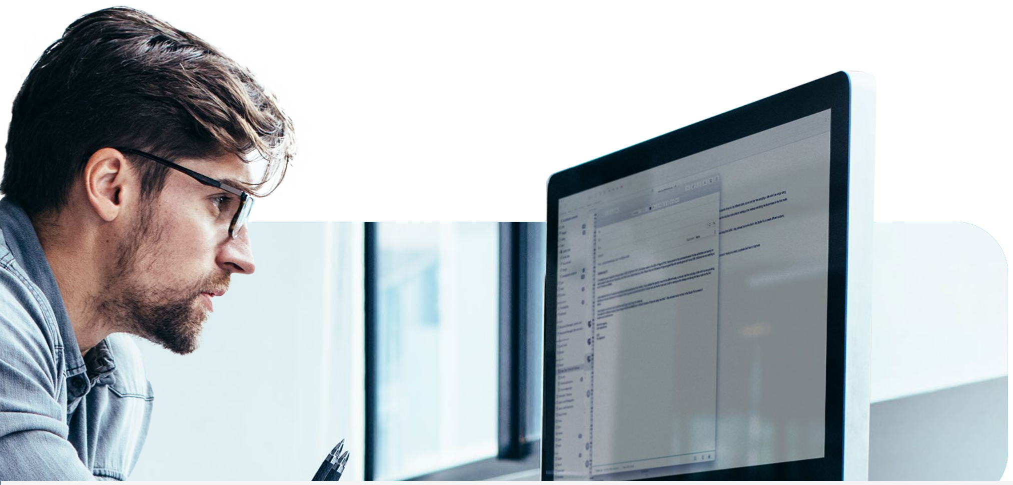
Architecting for **Agile DevOps**

We recently discussed the case for Agile Enterprise Architecture (EA), where the concepts of Agile are adopted into the working practices of an EA team. What we did not address was the perhaps more basic question of how an EA team can enable the adoption of Agile DevOps. These are two different questions, after all. There's no reason that an Agile EA team could not work alongside more traditional development teams if an organization wanted. Thus, in this paper we'll examine the architectural approaches to Agile.

What is Agile DevOps?

For those who are already familiar with Agile or have read any of our previous resources around Agile, feel free to skip this section. First, a note: Though you will often see the two terms paired together, Agile and DevOps are separate concepts that can be utilized individually; there is no requirement that agile teams also use DevOps or vice versa. However, in general the two are deployed together, and so you can assume that discussions in this paper can apply equally to Agile and DevOps.

Though often regarded as quite a new development, the Agile methodology has now been around for 20 years, after being first defined in 2001. Initially a simple statement of development principles, **it has evolved into the following 12 principles:** →



1. Our highest priority is to satisfy the customer through early and continuous delivery of valuable software..
2. Welcome changing requirements, even late in development. Agile processes harness change for the customer's competitive advantage. .
3. Deliver working software frequently, from a couple of weeks to a couple of months, with a preference to the shorter timescale.
4. Business people and developers must work together daily throughout the project.
5. Build projects around motivated individuals. Give them the environment and support they need and trust them to get the job done..
6. The most efficient and effective method of conveying information to and within a development team is face-to-face conversation.

7. Working software is the primary measure of progress. .
8. Agile processes promote sustainable development. The sponsors, developers, and users should be able to maintain a constant pace indefinitely.
9. Continuous attention to technical excellence and good design enhances agility.
10. Simplicity - the art of maximizing the amount of work not done is essential. .
11. The best architectures, requirements, and designs emerge from self-organizing teams
12. At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behavior accordingly.

As you can see, it is a software development philosophy, and this is where it has found the most use. Nonetheless, there have been attempts to adapt Agile to the wider organization, though it has not spread quite as well. In a more practical sense, the Agile philosophy aligns with a number of actions and activities that will be familiar to many. Perhaps the most well known are the Scrum framework and its Sprint iterations. A Sprint is a short unit of development, typically one or two weeks, with specific goals, while the Scrum is the overarching framework which aims to deliver value through transparency and adaptability. Scrum teams build products in iterations of Sprints.

Another popular Agile framework is Kanban. Kanban was originally part of the Lean methodology pioneered by Toyota, and literally refers to the colored cards that Toyota would use to direct resources around their factories. In Agile, Kanban is used to match development capacity to their current work-in-progress.

DevOps, meanwhile, outlines a process that aims to speed up software development by integrating the development and IT operations teams – hence the name. According to Microsoft: "Rooted in stability, consistency and planning, the DevOps culture seeks to identify new ways to improve and streamline processes. As a result, DevOps focuses on maximising effi ciency, identifying programmable processes and increasing automation."

DevOps and the various Agile frameworks are all aimed at improving the speed and responsiveness of software development and so it's no surprise that they are often paired together. According to IBM:

"Agile evolved the "big bang" approach into a series of "smaller snaps" which also compartmentalized risks. The more effectively these agile development practices accelerated software development and delivery, the more they exposed still-siloed IT operations as the next bottleneck in the software delivery lifecycle. So, DevOps grew out of agile."

Confl icts between Agile and EA

Given the above, it's easy to see where the difficulties arise with EA. Both EA and Agile encompass a wide array of different ideas, frameworks, and processes, but at a base level we can say that EA is focused on delivering medium and long term digital transformation, while Agile looks to short-term, incremental change that responds to customer needs.

EA places an emphasis on the architecture process and clear documentation, while Agile de-emphasizes those in preference of individuals, interactions and user stories. Similarly, Agile's implementation principle of welcoming changing requirements are directly opposed to the Enterprise Architecture approach. Agile also cites simplicity as a key implementation principle, but one of the key roles of EA is managing very complex systems. Enterprise Architecture is also typically focused on the enterprise itself and its key stakeholders, whereas Agile explicitly places the priority on the end customer.

Similarities Between Agile and Enterprise

Probably the two key similarities between the Agile approach to technology and the Enterprise Architecture approach are the focus on goals and outcomes, and the effort to simplify where possible. Both Agile and EA seek to rationalize unnecessary technology and eliminate ineffi ciency. Both approaches are also outcome driven, seeking to satisfy the end user and achieve the goals of the process.

Cooperation between business and development is a key motivator of EA and Agile, and both stress the need to continually keep assets up to date and relevant, whether focused on roadmaps and technology architectures or on products and services themselves. Sustainable development can also be seen in the goals of Enterprise Architecture. The creation of roadmaps and management of business capabilities are a form of insuring that the enterprise anticipates the future and can rapidly change to meet challenges. The primary goal is to enable a proactive response in place of a reactive one, but proactive addressing of changes and requirements is part of how sustainability is implemented.

How does EA help Agile?

By using Enterprise Architecture as a framework on which to implement Agile solutions, it is possible to integrate the two. Enterprise Architecture thus becomes one of the customers in an Agile development cycle – adding in requirements to both the Product and Sprint Backlogs. More generally, **EA can bring the following benefits to Agile:**

EA lays the Foundations for Agile

Agile projects begin with Iteration Zero, which lays out the scope of work and the setup of the project. High-level architecture can provide the foundation for iteration zero, helping to prevent projects from becoming siloed.

Focus on Value, not Infrastructure

Agile teams need to spend time on delivering improvements to end-users, not building out the necessary infrastructure for their development operations. EA can take care of infrastructure.

Adjusting EA for Agile

As mentioned above, "Agile EA" has become a goal of many fi rms. EA can be very flexible, easily allowing for multiple approaches and internal cultures, which make it simple to tailor EA to Agile.

Convert Business Requirements into Agile projects

Communication between business and IT is difficult; EA acts as a bridge between the two, taking high level business requirements and translating them into actionable insights for agile teams.

EA can Scale Agile

Agile DevOps is great for startups and small teams but doesn't prescribe any methods for scaling beyond that. Whether through a framework like SAFe or other methods, EA can help to enable agile strategies to expand across an enterprise.

Architects Align Agile Teams withEnterprise Vision

For enterprise-level work, new developments can often cut across multiple systems or create new systems; architects can ensure these developments continue to align with the vision of the architecture.

Practical Steps to Integrate EA with Agile

How do organizations achieve the above list of benefits, considering the conflicts that the two concepts face? The general outline of how architecture can help to deliver Agile should already be clear: EA should function as a connector, helping to convert business strategy into agile requirements and ensure that Agile teams remain integrated with the wider business and do not become siloed.

More specifically, EA in an Agile organization should involve much closer collaboration between architects and development teams. A centralized EA function would not work well in such a situation. Instead, Enterprise architects should be embedded directly in Agile teams, and if feasible some architects should be "Agile Architects" that can function as both agile development staff and architectural staff. A Lead Agile Architect sits at the top of the EA function, helping to facilitate Agile across the enterprise while acting as 'product owner' for EA, leading the creation of necessary architectures for Agile.

While people will play a huge role, EA also places a lot of emphasis on documentation. This can clash with the agile mindset. One way around this is to make use of user stories, converting traditional architecture documentation into architecture user stories, thus enabling guidance for Agile teams even without direct architect oversight. In Agile, the user story is an informal way of describing a product feature from the point of view of an end-user (often a customer). Thus, an architecture user story might be written from the PoV of a Solution Architect or other develop who implements architecture, and describe the features of said architecture in an informal manner.

One question that an EA team would need to answer is the position and role of Solution Architects in the Agile organization. In many respects, Agile teams can take over the traditional responsibilities of the solution architect. In one sense then, are solution architects required? Could they simply be rolled up into agile teams? Alternatively, could solution architects play a role in reconciling the developments of agile teams with the existing architecture? This question rests heavily on the position of the firm. An Agile company that is seeking to scale up with the help of EA is likely to take the former position, in which solution architects are somewhat unnecessary, whereas an enterprise hoping to adopt Agile may favor the latter.

Architects are also well positioned to take ownership of large-scale technology initiatives, often concerning the technology infrastructure that helps to enable Agile teams. For example, cloud migration could be a complex, large scale project that is poorly suited to an Agile team and best left to enterprise architecture. Agile teams can place a high demand on Architecture Review Boards (ARB), and so Gartner suggest switching to a just-in-time architecture assurance model to ensure that the ARB is able to meet the needs of the organization. This pivots the ARB from a rigid review schedule towards a fl exible system that is ready to review and recommend as and when Agile teams have the need.

Another Gartner suggestion is to move architecture away from document-based reference architectures, towards reference implementations. A reference implementation could be a pre-built virtual container or API library that developers can immediately deploy, rather than having to expend time to read and understand a reference architecture. This bakes architecture principles into development work without any barriers, preserving the speed advantage of Agile. Of course, this is only applicable to lower level architectures in the technology sphere, as you certainly can't deploy virtual containers to the business or strategy layers!

Summary

In this eBook, we have looked at how enterprises can adopt both Agile and Enterprise Architecture without causing conflict or inefficiency. First, we discussed the definition of Agile and the often paired concept of DevOps, identifying the 12 guiding principles of the Agile philosophy. From there we were able to detail how Agile and EA are similar and how they are different, with time horizon and scale being the crucial issues.

Despite these differences, we detailed 6 key benefits that could arise from the integration of EA and Agile, noting that EA can help prevent Agile teams from becoming siloed, and align their development with wider business goals. Finally, we looked at some practical advice for Enterprise Architects and the EA practice, highlighting suggestions such as architecture user stories and agile architects. What is hopefully clear is that there is no reason Agile and EA cannot both play important roles for organizations, and each one can enhance the other.

Enable Agile DevOps In Your Team

Book a tailored demo today to find out how the iServer Suite delivers faster, smarter decisions that enable transformation

Book a Demo

OrbusSoftware iServer365

OrbusSoftware