Accelerating Application Delivery in a Hybrid World

● James Weir ● Alban Richard ● Yuzuru Ueda

Today's businesses are looking to increase ICT responsiveness with agile methodologies and DevOps processes. But they must also ensure that governance and compliance are not compromised. Furthermore, hybrid ICT or multi-cloud environments can exacerbate these conflicting requirements, where different infrastructures are used for development, testing, pre-production, and production environments. DevOps can be interpreted in many ways: there is no common definition. It can be seen as being an environment that promotes communication and collaboration, while Agile is a method of working within the environment. DevOps stresses effective collaboration and communication between various teams and departments within a culture that optimizes release cycles of high-quality and thoroughly-tested end products. By viewing the entire delivery process holistically, DevOps helps us identify and solve bottlenecks that traditionally happen when one role in the process is overloaded. UForge AppCenter is a platform that follows DevOps principles and focuses on reducing manual coordination across the different stages of the application delivery life cycle to speed up the process end to end. UForge provides native hybrid functionality to DevOps processes. Cloud-neutral application templating enables repeatable processes that can be used across multiple clouds or datacenters throughout development, testing, pre-production, and production. This paper describes how UForge can provide the speed and agility needed from DevOps without sacrificing control and consistency.

1. Introduction

With the pace of business becoming faster than ever before, businesses must evolve and adapt to meet new customer needs with more agile and responsive ICT without compromising governance and compliance.

While cloud computing helps provide infrastructure on demand, building and deploying applications often remains a bespoke and largely manual process, which has become an inhibitor to both agility and governance.

Despite the tools already in place, application deployment in a hybrid environment can become a challenge, especially when manually integrating builds from scratch for each infrastructure. Nowadays, developers are looking to automate that process, ensuring consistent, repeatable application builds regardless of infrastructure while ensuring full software governance.

This paper describes Agile development,[1)] how it relates to DevOps, and how UForge can provide the speed and agility needed from DevOps without sacrificing control and consistency.

2. Agile development and DevOps

To better understand the nature of Agile development and its relation to DevOps, one of the authors attended a workshop on Agile development to learn about what kind of mindset is required to create a product that would satisfy a customer's needs.

2.1 The workshop challenge

One of the challenges of the workshop was to work as a team to reproduce a drawing (**Figure 1**) provided by the instructor (playing the role of the "customer"). The group was divided into three teams of four people. Each team included two "designers" and two "artists" who needed to collaborate to reproduce the drawing. The designers were shown the picture, which contained a set of shapes and patterns. Their job was

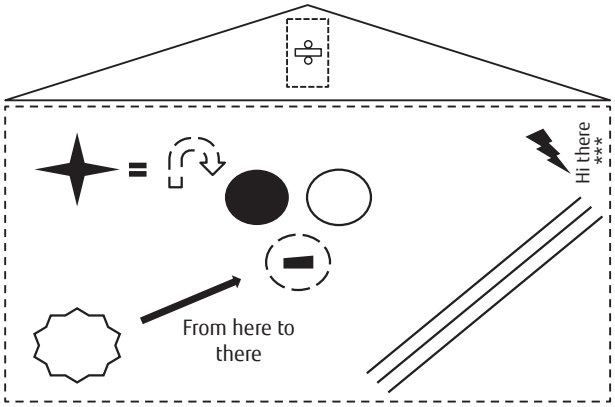

Figure 1
Original drawing to copy as part of Agile workshop.

to describe what they saw to the artists who then had to re-draw the picture from the designers' instructions, all within ten minutes. However, the designers were not allowed to speak to the artists. The author was asked to be an artist.

The workshop instructor took the designers out of the room to show them the picture, and the timer started. After about six minutes, the designers from the other teams started to arrive with a series of post-it notes covered with instructions, which were handed to the artists. The designers then quickly left the room. After eight minutes, the designers from the author's team appeared with an A4 page full of detailed instructions. The author's team, we had only two minutes left to follow the instructions and complete the drawing. The following gives an idea of the written instructions on the page:

- Draw a rectangle taking up three quarters of the page, dotted lines
- Draw a triangle on top of the rectangle
- Draw three circles in the middle

We were able to complete 20 percent of the instructions by the time the ten minutes were up. Unfortunately, when the instructor showed us the original picture, we discovered that our picture, as well as those of the other teams, did not bear any resemblance to it.

The goal of this exercise was to demonstrate certain types of behavior when working in a team. First, as soon as we were separated into designers and artists, the team dynamic changed. The designers were responsible for noting down the various shapes in the picture and the artists had to try to re-draw them.

Second, the artists only started receiving requirements from the designers after six to eight minutes had already passed, leaving only a few minutes for the artists to work.

Finally, every team used the same method; a designer would arrive with written instructions on a post-it note or piece of paper, leave that with the artist, then immediately leave the room again to collect more data. The artists were left to themselves to try to understand the instructions and execute on them.

From these three observations, it was clear that the team was no longer working cohesively, but as two sub-teams. The designers would do their job by noting down the shapes, then hand over the responsibility to someone else to finish the work i.e. the artist who now had an immense amount of pressure to finish the task in the allotted time.

Rather than the team taking collective responsibility for the entire task, the responsibility was divided, and a natural barrier –in other words, a silo– was created. It took too long for the artists to receive the first piece of information, and it was impossible to gather any feedback on their interpretation of the instructions as the designer was pre-occupied with looking for additional data.

Once the picture was shown, the instructor asked how long it would now take to complete. Looking at what we had done raised the question of whether the shapes we had drawn were within the customer's requirements and how long it would take to either restart or complete. We discovered that, during the exercise, none of the designers had thought to ask the instructor (the customer) if any of the shapes were of higher priority to draw than others.

The way in which the teams had worked was very much like the traditional Waterfall method for software development.

In Waterfall development, software applications are delivered in a linear way, passing through different teams in the process. Each team has certain responsibilities that need to be completed prior to handing the application off to the next team. Project managers (our designers) receive requirements from various sources, and the development team (our artists) interprets those requirements then develops the software application. The application is then sent to the testing team to be qualified. Once qualified, the release engineering

team packages the product, which is either delivered to the customer or handed to the operations teams to deploy it in production where it is monitored.

The goal of the Waterfall model is to minimize risk, and that is where the problem lies. Using the Waterfall method of development slows down feedback, which requires checkpoints for teams working in isolation for each iteration of the product. Development teams only have one chance to get each aspect of a project right.

2.2 Meeting the challenge: Agile development

Agile development prioritizes maximizing agility rather than minimizing risk. It limits the scope of a project or product feature by setting a minimum number of requirements and turning them into a deliverable product, and focuses on concepts such as "fast and efficient; small; lower cost; fewer features; shorter projects."[2)]

With this information, we repeated the exercise with a new picture. The results were significantly different. When the designers left to see the new picture, the first thing they asked the instructor was which shape had the highest priority. Within 30 seconds, all of the designers were back with a simple instruction on which shape to draw. Rather than leave the artists with the instruction and go back for a second shape, they watched the artist begin to draw the shape. They would then write small notes (as they were still not allowed to talk) to provide feedback and improve the shape (for example "left a bit," "a bit bigger," etc.) This helped ensure the artist was drawing the shape at the correct size and in the correct position before going back for a second shape.

In this way, the teams broke down the large task of drawing the entire picture into smaller tasks, executing on them, providing feedback, and completing that task before tackling the next one. Even though we did not completely finish the picture, we had completed the picture with the most important shapes. In other words, we had a minimum viable product the customer was happy with.

2.3 Relation to DevOps

DevOps can be interpreted in many ways. In the authors' opinion, DevOps is an environment that promotes communication and collaboration, while Agile is a method of working within the environment. DevOps stresses effective collaboration and communication between various teams and departments within a culture that optimizes release cycles of high-quality and thoroughly-tested end products.[3),4)]

This release cycle, or software delivery process, can be divided into seven phases shown in **Figure 2**:

- BUILD: This is the development stage of an application. The development teams work on new features, enhancements, and bug fixes.
- TEST: While an application is being developed,

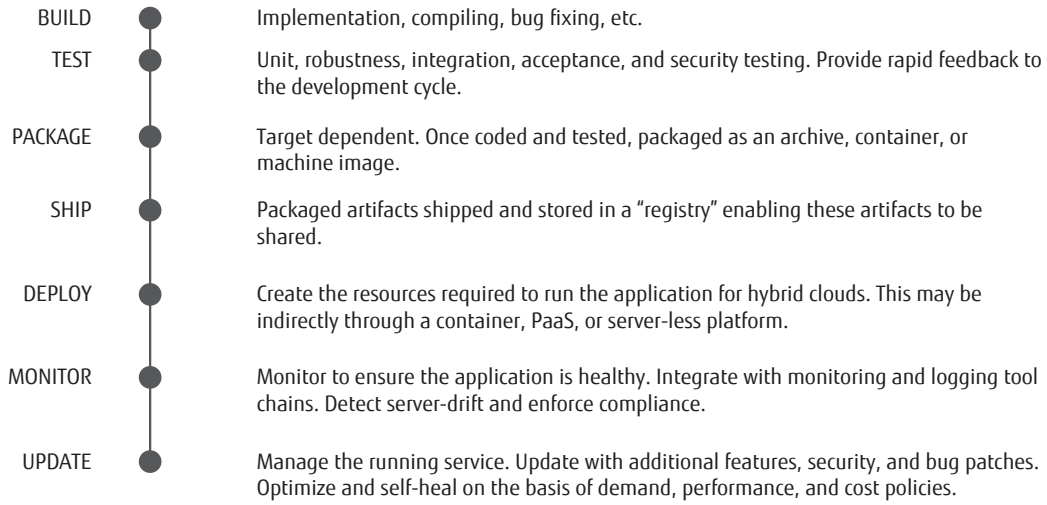

Figure 2
Modern application lifecycle.

it will be tested. Unit tests are usually written as part of the development phase, but can be supplemented with integration, security, performance, acceptance, and scenario testing. Testing is critical in ensuring new developments do not add regressions and providing feedback on the current quality of the features. Testing usually involves both development and operations teams, but may also involve a dedicated quality assurance (QA) team.

- PACKAGE: Once an application has been tested, it must be packaged correctly to enable it to run on the chosen target environment. The packaging depends heavily on the technology or environment. For example, this might be a WAR file for Tomcat, an OVF image for VMware, or a Docker machine image for Kubernetes.
- SHIP: Once the application has been packaged, it needs to be made available for other teams, partners, or customers to consume. A registry is used for this purpose. The type of registry depends on the package type used. For example, a Docker machine image may be pushed to DockerHub, and a WAR file added to Nexus or Artifactory.
- DEPLOY: Applications require somewhere to run. Under the layers of abstraction, there are still computation, network, and storage resources that must be provisioned, and the application must be "installed" either from a machine image template or another artifact.
- MONITOR: Running applications need to be monitored to ensure they continue to run correctly and are healthy. Configuration monitoring may also be used to ensure there are no misconfigurations of the system or to detect unauthorized changes.
- UPDATE: Once an application is deployed, it must be updated over time with security patches or the latest features. Eventually, it will be decommissioned when no longer required.

Since the seven steps of delivering an application cannot be skipped, the priority must be how to minimize the execution of each step. If each team is empowered to work independently, then coordination can be reduced and individual productivity can be increased, and consequently, application delivery velocity increases. This is the essence of DevOps. To maintain a consistent process, any tools chosen must focus on workflows to address the technical heterogeneity that is the reality of most organizations.

3. DevOps with UForge AppCenter

UShareSoft, a French software development company founded in 2008 to simplify application delivery, was acquired by Fujitsu in 2015. The company is now part of Fujitsu's Platform Software Business Unit. Its flagship product, UForge AppCenter[note)] has been designed to automate some of the seven steps of the DevOps release cycle, helping to accelerate application delivery in hybrid environments (**Figure 3**).

note) UForge AppCenter is sold as "FUJITSU Software UForge AppCenter" by Fujitsu. Since 2015, it has been jointly developed and sold by both UShareSoft [5)] and Fujitsu.[6)]

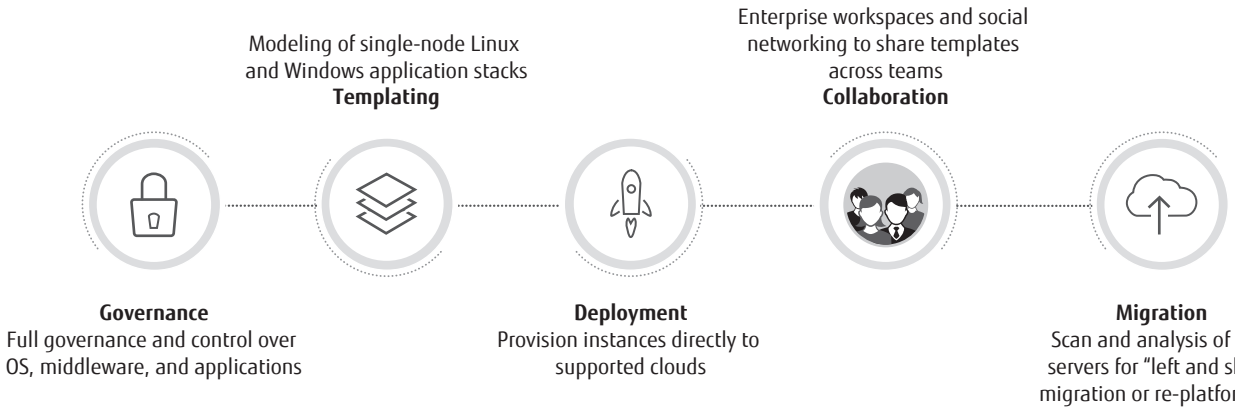

Figure 3
UForge for application delivery.

Figure 4
UForge coverage for application delivery.

It enables developers to model the complete software stack (OS packages, middleware, application, and configuration information), then automatically package it to run in any infrastructure, including Fujitsu Cloud Service K5, Amazon Web Services, Microsoft Azure, VMware, Docker, etc. This cloud-neutral application templating (known as "application as code") is unique to Fujitsu, and combined with automated delivery processes, dramatically accelerates application release cycles in hybrid environments.

Application templating also enables UForge to bring software governance into enterprise DevOps processes by ensuring full control of applications. UForge models;

- Native package repositories, tracking updates, providing search features, and ensuring that packages are kept synchronized with repositories and updated correctly.
- The full software stack including low-level operating system parameters (keyboard, time zone, partitioning, etc.) and operating system package dependencies, as well as components further up the stack: middleware, applications and configuration logic. This enables transparency into the full stack and consistent deployments across clouds.

UForge works with other DevOps tools, including continuous integration and delivery tools, to create an automated, repeatable enterprise DevOps toolchain from coding, building, and testing, to release and deployment (**Figure 4**). The UShareSoft team is continuing to innovate with new modules in the DevOps space, which includes leveraging open source software, the latest being Apache Brooklyn[6] for open standard hybrid ICT blueprinting (**Figure 5**).

4. Application migration with UForge AppCenter

As well as DevOps, UForge can also automate application migration to enable portability between platforms and ensure customers are not locked into specific clouds or vendors. Enterprises can simply "lift and shift" servers for fast migration with minimum disruption. Alternatively, users can create a template from live servers, enabling them to refactor applications during migration to improve governance, life cycle management, performance, and other benefits in the cloud.

5. Conclusion

This paper described Agile development, how it relates to DevOps, and how UForge can provide the speed and agility needed from DevOps without sacrificing control and consistency.

By viewing the entire delivery process holistically, DevOps helps us identify and solve bottlenecks in application delivery that traditionally happen when one role in the process is overloaded. UForge focuses on reducing manual coordination across the different stages of the DevOps life cycle to maximize the velocity

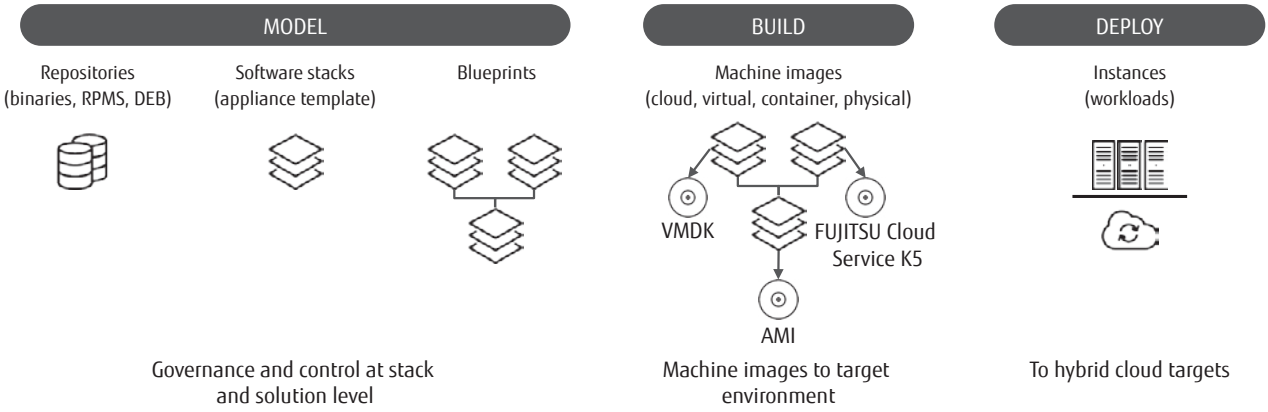

Figure 5
New blueprinting module.

of software delivery.

UForge also brings native hybrid ICT functionality to DevOps processes. Cloud-neutral application templating enables repeatable processes that can be used across multiple clouds or datacenters throughout development, testing, pre-production, and production, bringing governance and control at the same time as speed and agility.

References

1) Agile Alliance: What is Agile Software Development? *https://www.agilealliance.org/agile101/*
2) CodeProject: An Introduction to Agile Methodology. *https://www.codeproject.com/Articles/704600/An-Introduction-to-Agile-Methodology*
3) Agile Buddha: Demystifying DevOps: Difference between Agile and DevOps. *http://www.agilebuddha.com/agile/demystifying-devops/*
4) Atlassian: Agile and DevOps–Friends or Foes? *https://www.atlassian.com/agile/devops*
5) UShareSoft: Automate Cloud Application Delivery to Support Your Digital Transformation. *https://www.usharesoft.com/*
6) Apache Brooklyn. *https://brooklyn.apache.org/*

James Weir
UShareSoft, SAS.
Chief Technology Officer

Alban Richard
UShareSoft, SAS.
Chief Executive Officer

Yuzuru Ueda
UShareSoft, SAS.
Project Manager