TechData
Global Lifecycle Management

ExitCertified®

Transforming Software Development in the Enterprise:

AGILE, DEVOPS AND KUBERNETES

WHAT'S INSIDE

THE GROWING NEED FOR TRANSFORMATION

Advances in development techniques — first with Agile and now DevOps — help enterprises deliver software applications faster and with greater reliability. And the addition of container-based deployment along with Kubernetes orchestration makes it easier to leverage cloud features to build more portable, sustainable software.

These advances help enterprises keep up with the ever-increasing speed and complexity associated with cloud environments. Yet even with clear indicators of the benefits that can be gained from new methodologies, many organizational leaders find that it's not always easy to transform from legacy development practices.

The adoption of new methodologies requires a combination of upskilling and cultural transformation. But in the face of constant deadlines and competing priorities, development teams often resist the necessary changes.

Transformation is necessary in order to stay competitive in today's market, and many organizations are already capitalizing on modern development practices and utilizing training at all levels of the organization to help break down barriers and transform their enterprise.

THE LEGACY ENTERPRISE DEVELOPMENT ENVIRONMENT

Through the legacy waterfall development process, the business determines there is need for a piece of software and then develops specs and sends them to development, who then generates the code and sends it to operations to deploy. This process is time-consuming and serial in nature, focusing on the creation of a fully featured and rigorously tested application, and can take months, if not years, to complete.

In these legacy environments, each software release presents a high-risk, high-reward moment. All too often, software products experience unforeseen defects in production, or the operational systems prove to be inadequate for the user load. Additionally, the long development cycles allow for significant changes to requirements and scope creep, which prolongs development cycles and pushes release dates to the detriment of the company's revenue streams.

By the time software gets released, the long development cycle has allowed for competing products to emerge on the market. In a business environment where the first to market is often the market-share winner, product development cycles have a significant effect on company revenue streams.

With the widespread adoption of cloud technology and ever-increasing consumer demands for more and more features, most successful organizations have abandoned the waterfall development process due to its slow and problematic nature.

ACCELERATING DEVELOPMENT PRACTICES WITH AGILE TECHNIQUES

Most organizations today employ Agile techniques that use an iterative process for software development. The Agile approach involves all stakeholders who are going to build and deploy the software at the outset of the project. They develop a minimum viable product and then add to it iteratively as development proceeds in short sprints. The first production release occurs once there is a functional application that offers value to the customer, even if it's not yet fully featured. The Agile process breaks up big, complicated projects into much smaller and more easily managed pieces, allowing developers to work faster on a product that can go to market sooner.

Agile mitigates the inherent risks and steep costs of legacy waterfall processes by starting small and building in iterative sprints. With each sprint lasting a short duration and producing a limited and controlled set of features, you can validate your strategy in a production environment, learn as you go and update priorities to meet customer-driven requirements. As the project proceeds, risk reduces with each sprint, and each new feature adds to the robustness of the solution. The regular pattern of delivery also yields more predictable and controllable costs.

Additionally, using Agile development techniques can lead to:

Improved customer satisfaction

- Because the process is iterative, the business can take feedback into consideration and effectively build a product that balances the needs of the business with those of the customer.

Better project governance

- Agile reduces project management to well-defined sprints while also clearly defining a backlog for holding new or changed requirements. This process gives project managers greater control and, when effectively managed, eliminates scope creep.

Faster return on investment

- Not only is the overall development cycle shortened, but incremental development also means customers can start using — and paying for — the application while it is still in active development.[1]

THE LIMITATIONS OF AGILE

Agile development solves many of the problems of the waterfall method, yet in today's cloud architectures and nonstop technology transformations, even Agile can be too slow and cumbersome. Often, conflicting goals and approaches create a barrier between development and operations teams. The development team's goals center on innovation and rapid deployment while the operations team's goals focus on ensuring that that the software runs in the right environment, can be accessed by the right users and that the environment stays up and running to meet pre-set service-level agreements (SLAs).

The point of heaviest friction between these two groups is often software release. In Agile, the development team is rewarded for changing the system, while the operations team is rewarded for keeping production application stacks and infrastructures running. Change brings risk, and the two teams' goals conflict at the moment of production release. While operations teams have systems in place that control deployment, those systems can slow down the development cycle. And with Agile development teams targeting multiple release cycles per month, those processes represent a risk to delivery.

DEVOPS: BREAKING DOWN THE BARRIERS

There is a better way: DevOps — which is a strategy that forms cohesive teams with representatives from both development and operations.

With the integration of development and operations teams, the DevOps strategy helps to resolve friction between organizations, accelerates delivery and improves quality by breaking down barriers and implementing a system of shared goals and objectives. It is both a cultural shift and a set of practices and tools that increase an organization's ability to deliver applications and services at high velocity. Two of the most important benefits of using a DevOps approach are:

Increased focus on automation

Automating deployment can help operations teams implement software products and release services up to 30 times faster than legacy processes. But deployment is only one small facet of automation across the DevOps toolchain. By automating environment creation, software builds and testing, security testing, software deployment and delivery and production issue reporting, DevOps teams create a continuous integration/continuous delivery (CI/CD) pipeline that automatically pushes tested, validated and secure code to production or production-like environments without human intervention. This reduces the strain on both development teams and operations engineers so that they can focus on future deliverables.

Mutually shared goals and deliverables

When development and operations teams work together in a DevOps environment, they are working toward shared goals and deliverables, with each team taking ownership for their part in overall product success. In DevOps, neither team can succeed without the other, so it fosters an environment of collaboration. This also puts the focus on the product rather than the team or the individual. Developers and operations engineers end up with a holistic understanding of what the product needs for success, rather than focusing on myopic deliverables and laborious manual processes.

DEVOPS AND THE NATURAL PROGRESSION TOWARD CONTAINERIZATION

The microservices architecture breaks applications into self-contained pieces of software that are deployed independently and are extremely fault-tolerant. Instead of building one, large executable that dictates everything an application does, you build little executables and stitch them together. With Agile and DevOps techniques, development teams can produce their applications in manageable pieces that take operational needs into account, and the operations teams can deploy those application pieces faster, with less apprehension of adverse impact on their systems.

In a microservices architecture, the application is made up of separate, smaller services that have multiple entry and exit points with clearly defined APIs. They operate independently, and when running in a containerized environment, each container maintains its own operating environment as well (including any or all: CPU, GPU, memory, storage, databases, connected services, etc.). They are all distributable and reusable, facilitating agility in design and yielding more scalable and fault-tolerant applications. But this does create a problem with networking because you have to keep track of all those services. This could increase latency, but properly deployed microservices with good orchestration can render latency negligible. Operations teams do have to contend with managing storage requirements, load balancing across the network and handling an increase in the number of processes running all those little software services.

Containers help to mitigate these problems. You package up each individual piece of software in a microservices architecture, along with all of its dependencies, into its own separate container. This isolates each piece of software from all the other pieces of software and from the hosts they run on. Isolating software pieces is one of the primary benefits of containerization, as it allows the software pieces to run correctly and reliably in different computing environments. It decouples the development environment from the operations environment. And if your servers are running Docker — the most popular container platform — when you launch a container, it gets downloaded and runs. When you stop it, it is gone. Containers don't get installed, so there is no lasting effect on the OS.

Transitioning your organization to DevOps can be a significant departure from existing practices and should be carefully planned out. Before expecting existing teams to dive into using modern DevOps practices, they'll need at least some foundational knowledge. Learning on the job introduces risk, and you could end up with a traditional application stack running inside a container. You could start a new DevOps project and focus on automating the Agile environment, or you could start by migrating a legacy application to a containerized environment. The most important first step, though, is to train your teams.

KUBERNETES AND CLOUD-NATIVE DEVELOPMENT

Microservices architecture can quickly lead to unsustainable complexity — that's where Kubernetes comes in. Kubernetes provides horizontal scaling, load balancing and resource management for containerized workloads. It is open source, maintained by the Cloud-Native Computing Foundation, and is the most popular container orchestration program in use today. Although Kubernetes is often called a platform, it is more accurately a framework. You run your microservices-architected application on a Kubernetes cluster and let it manage the containers, networking and storage requirements. Applications are run in pods that include all the containers required to run the application, plus configuration, storage, networking and resource management. Pods are transient; Kubernetes invokes instances of the application as required.

Kubernetes is API-driven to facilitate DevOps and integration with cloud providers. It is unopinionated, so you do have to add centralized logging and monitoring, but it is more flexible than opinionated frameworks. It may take a bit longer to ramp up when you are first learning about Kubernetes, but the better flexibility downstream is well worth it.

Developing your software applications using containerization and Kubernetes orchestration makes your application cloud-native, which simply means that the application takes advantage of the features of the cloud. Cloud-native principles include:

- ✔ **The application is developed in a microservices architecture**
- ✔ **Each microservice runs in its own process and communicates via APIs**
- ✔ **Microservices are designed to be fault-tolerant and scale horizontally**

ADOPTING NEW TECHNIQUES REQUIRES ORGANIZATIONAL CHANGE

New techniques are available to help your organization develop and deploy the highest-quality software for your clients while controlling costs. But adopting them means abandoning, to some extent, business as usual. It requires a change to both culture and processes, but enterprises tend to resist major disruptions. Leadership as well as technical personnel in all the participating teams — business, development and operations — must embrace these new practices to see transformative results.

In addition to requiring a culture change, moving from a legacy waterfall approach to an Agile or DevOps approach requires a sound foundation of training. DevOps training helps you gain critical expertise in the principles of continuous development and deployment and also provides knowledge regarding the tools and techniques needed to apply those principles to achieve inter-team collaboration and maximum benefit for your organization.

ExitCertified helps organizations make the migration to DevOps through training from the business and organizational perspective, as well as the enabling of technologies such as microservices, containerization, network design, Kubernetes orchestration and automation/configuration management training. When a leading financial services company needed to migrate from a large enterprise-class platform-as-a-service to a Kubernetes-based solution, it turned to ExitCertified to upskill its IT professionals across a range of cloud and microservices competencies. The training they received helped accomplish two major initiatives within an expedited timeframe.

Read the Case Study

PARTNER WITH **EXITCERTIFIED**

ExitCertified provides a full range of training, and you can take the classes most appropriate for your needs based on your job role and level of experience. When you train in ExitCertified facilities, you will find well-equipped classrooms and expert staff who are dedicated to making your learning experience comfortable and productive. And when you train remotely, ExitCertified's unique Individual Multimedia Video Presence (iMVP®) training platform makes online learning every bit as engaging as in-person training.

TechData Global Lifecycle Management | ExitCertified®

To learn more, visit exitcertified.com and explore the training options available for Agile, DevOps, Kubernetes and microservices.

Source

1 **Lucidchart**, "Agile operations 101," https://www.lucidchart.com/blog/agile-operations-definition