Accelerating Deployment of Pay TV Systems with DevOps

Avraham Poupko – apoupko@cisco.com

What Is DevOps

Traditionally, organizations that develop software (such as Cisco, software vendors, and internal development at large service providers) have an engineering or development unit ("dev") and an operations unit ("ops"). These two organizations engage with each other and with their customers in a rhythm set by the release cycle.

The unit of delivery is a release. The engineering unit develops the system and implements features based on specifications it has received from the customer. After engineering completes development, testing, and quality assurance, it packages the new system into a release and passes that on to the operations unit. Operations configures and deploys the release into its own internal environment (and also might conduct its own testing) and subsequently deploys the release to the customer for use in a production environment.

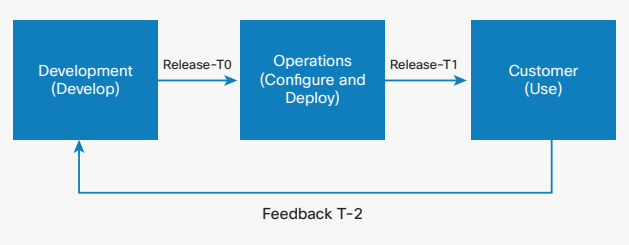
The customer evaluates the system, and if it is found to perform to the customer's specification, the customer accepts the release. This includes deploying it into a real production environment and paying for it as agreed in the contract. After the customer sees the release in operation, the customer provides feedback and requests for changes and updates, and these are provided in the subsequent release from the engineering/operations units. The process is rather linear.

One of the main differences between development and operations is in their environments. The development environment is typically appropriate for development functions. It includes compilers, debuggers, editors, code profilers, and other tools. Operations, in contrast, has an environment that is as close as possible to the actual environment in which the system will be deployed. This includes a database with data that is as close as possible to the data used in production. The environment also includes the actual hardware that is used in production as well as other tools such as loggers, load balancers, and the likes that are used in a real environment.

The main advantage of this method of operation is that it allows each unit to develop expertise in its respective area and to focus on what it does best.

This is how software has been developed for many years. (See Figure 1).

Figure 1. Traditional Software Lifecycle: Time between T0 and T2 Can Be Many Months



However, this traditional method also has several challenges. It has often resulted in customers not getting the features they want or not getting the features they want in time to bring business value:

- **Long cycle time for feature release:** A lot of time passes between the moment the customer imagines the feature and the moment when the customer is actually able to see that feature working the way it was originally imagined. The handovers themselves take time, and the feedback loop takes time. While a release is being tested in a production environment, it is not possible to continue work on that release in the development/operations environment. The differences in the environment also have the effect that releases have a certain overhead associated with them. The release provided by development has to be installed and configured in the operations environment before it can be used in the customer environment. As a result, the customer gets the feature months after asking for it, and if the feature is not exactly what the customer as asked for, the customer again waits months to get any updates to that feature. It is not uncommon for a year or two to pass between the time the customer first asks for the feature and the time the customer is able to deploy that feature in production.
- **Feature release that misses customer expectations:** The situation is made worse by the fact that development rarely gets the feature right at the first attempt. The engineers who develop the systems are often disconnected from the actual use of the system. Often their only understanding of what the system should do is based on the specifications they have been provided by the customer. Those specifications do not and cannot tell the full story, because the full story requires the context and details present in the customer environment. Because developers often have only the specification with which to work, they build a system that meets the specification and then test it against an acceptance test plan (ATP) that was derived from that specification. But that system is often not the system that the customer imagined. As a result, the feature goes through several rather long iterations to get corrected specifications from the customer.
- **Feature release that does not meet production quality:** The difference in environments also contributes to the disconnect. Often development delivers a system that worked just fine in the development environment, but does not work in the production environment, because development does not have the infrastructure, tools, and data mirroring of the production environment. This applies to functionality as well as to quality. Certain functions can be properly evaluated only in a production environment, and certain aspects of quality ("nonfunctional requirements") are put to the test only in a real production environment.

DevOps attempts to change all that. DevOps is a set of practices (supported by tools and culture) that attempt in enhance the communication around technology to provide for more frequent deployments and deployments of quality that meet and exceed expectations (functional and nonfunctional). DevOps is the practice of operations and development engineers participating together in the entire service lifecycle, from design through the development process to production support.[1] It allows for much shorter development cycles and for enhanced communication between the development teams and the operations teams. A primary corollary to this is that part of the major change in practice from previous methods is that DevOps is characterized by operations staff making use many of the same techniques as developers for their systems work.[2] The end result is that customers get the features they expect, and rapidly.

The basics of DevOps:

- The developing engineers have an environment that is as close as possible to the production environment.
- Every time the system is built, it is immediately tested within a productionlike environment, and only if that system works properly is it sent to production.

Taken to the extreme, DevOps allows a software vendor to provide a continuous value stream. No longer does the release set the cadence. Rather, the customer gets a constant stream of working features.

[1] Ernest Mueller – https://theagileadmin.com/what-is-devops/

[2] Ernest Mueller – https://theagileadmin.com/what-is-devops/

If a feature does not work properly in the deployment environment, or if the feature does not work as imagined, the feedback loop is short and fast,and the development team along with the operations team can perform the required fixes and quickly redeploy.

This is where Cisco is headed.

The Value of DevOps to Pay TV Operators

When properly implemented, DevOps brings the following advantages:

- The system is always working and always up to date. Whenever a new service is deployed, the operator knows that the service has been tested in an environment that is as close as possible to the real production environment.
- The system gets new features quickly. The time between the specification of a feature and its delivery is very short. When a feature is delivered, probably it is still relevant and useful. Operators are not told that they must wait for the next release in order to get a requested feature.
- Feedback cycles are short. If a feature is not properly implemented, does not work as imagined, or does not work properly in a production environment, the problem is discovered early on in development, and the feature can be modified without having to wait for a next release.

Another benefit of DevOps is real-time analytics. Development engineers are able to see real-time usage data of the features in the system, which allows the developers to identify bottlenecks in the software and respond to them. This allows for better focus of development resources. (It also requires a new set of tools that the developers can use to access this real-time data.)

Looking at the challenges outlined earlier (long development cycles and features that do not meet expectations of function and quality), it is easy to see why DevOps can address these issues and provide for greater customer satisfaction.

DevOps at Cisco

Cisco is continuously evaluating and adopting new practices to improve the way we develop software for our customers. Cisco is currently immersed in an agile transition that is allowing us to develop better and faster software.

As part of the transition, we have adopted DevOps. However, we did not just take DevOps as is. Rather, we have customized our own version of DevOps that is appropriate for the way we and our pay TV customers work.

The pay TV systems we deploy have two main characteristics:

- Our systems are composed of many subsystems that work together to provide a solution for a customer.
- Our systems often require custom development in order to address the exact needs of a customer.

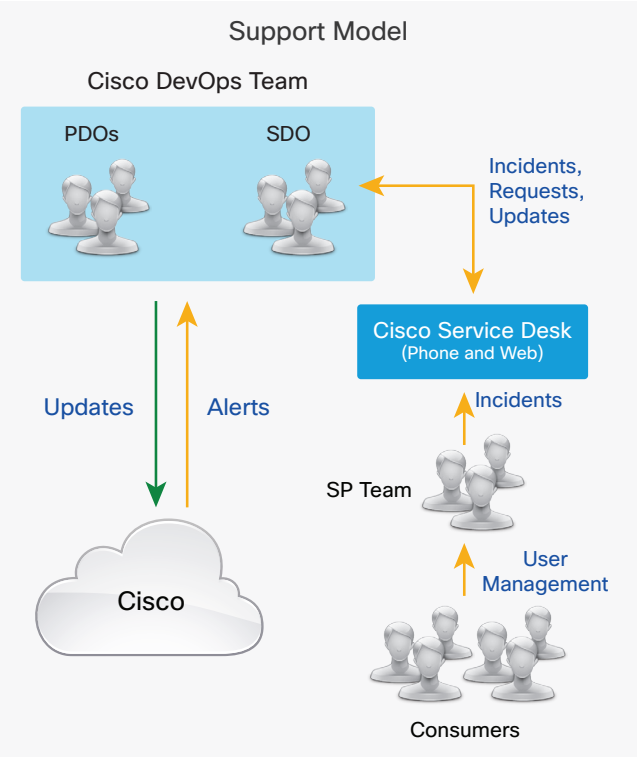
These two characteristics forced us to create our own variety of DevOps. Therefore, we have two types of DevOps teams.

Before we go on, some terminology:

- A product is a generic system that has a feature set and a roadmap. The product is defined by the product code.
- A particular deployment of a product or set of products is called a solution. The solution includes a full environment and configuration.

At Cisco, the DevOps cycle is supported by two teams, as shown in Figure 2.

Figure 2. DevOps Support Model



The product DevOps team (PDO) does continuous deployment and integration of the product into a "vanilla" environment. This vanilla environment is a full production environment that attempts to be based on the lowest common denominator shared by all potential customers.

The solution DevOps team (SDO) then takes the product from that vanilla environment and deploys it in a customer environment.

The customer provides feedback to the SDO. If possible, the SDO makes modifications to the configuration, to the environment, or even to the code, and redeploys.

The SDO also provides this information to the PDO, so that the product gets appropriately updated.

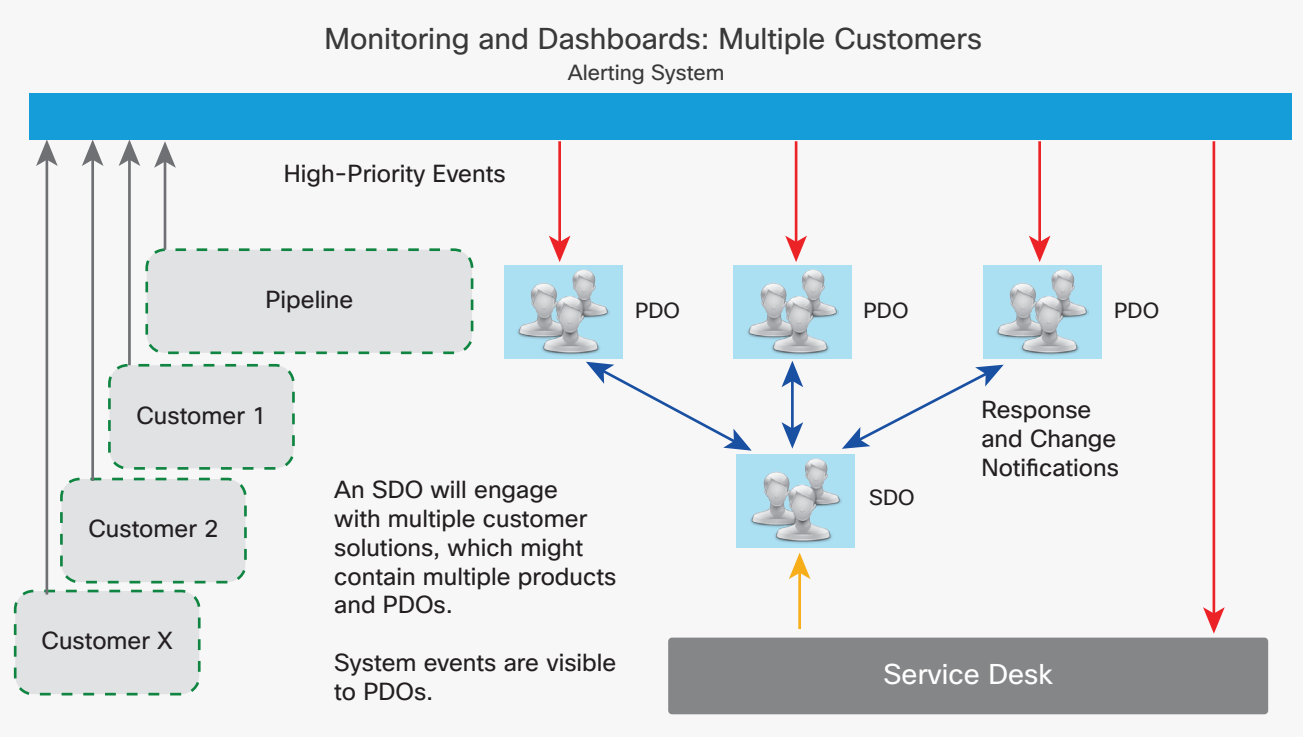
The advantage of working this way is that we can achieve a certain degree of specialization, while at the same time having a shared understanding and environment. The SDO understands the entire context of the solution and understands how all products work together to provide the solution, whereas the PDO understands the product in depth. Both teams work on the same code base and operate in the same type of environments.

DevOps for Multiscreen Pay TV Systems

This chapter includes an example of how DevOps was used to implement certain features in our multiscreen video solution deployed by pay TV operators.

Figure 3 shows the DevOps cycle.

Figure 3. DevOps Cycle



Our multiscreen video solution is a large and complex family of products. Each product has many teams of developers working on the various features. Each of the products has a PDO that is constantly making sure that all these features are working together in a generic staging environment that we have at Cisco.

The SDO then collects all these working products and puts them together into a working and configured solution that is appropriate for a customer.

One of our customers requested a particular kind of integration with a recommendation engine. The customer had requested that the recommendation engine take into account the content that the viewer had viewed across all devices, not on just one device. This required that the recommendation engine be aware of user identity in addition to being aware of device identity.

Implementing a DevOps practice, we were able to quickly provide an implementation to this customer. The PDO was able to make sure that this extra feature worked and did not break anything. Rapidly following, the SDO had an upgraded product in the context of a full solution as deployed for that customer and was able to see that this feature worked in the context of a real productionlike environment. Because all environments were similar, we knew that we could deploy it in the customer environment with confidence. We deployed it to the customer and used our real-time analytics capabilities to capture usage data to help confirm that the feature was well understood by the end users and was performing as expected.

We have not totally eliminated the concept of a scheduled release. We still have periodic releases to the customer. However, we have reduced the time between the point the customer conceives of a feature and the time that feature is delivered.

The Role of the Pay TV Operator

Like all agile practices, DevOps requires cooperation and involvement of the pay TV operator. This requires technical accommodation as well as cultural accommodation. On the technical side, pay TV operators need to be willing and able to accept continued upgrades in their environment and to not be bound by the cadence of releases.

This means an environment and skillset that support quick and easy deployment and configuration as well as allowing updates to the running system without affecting day-to-day operations.

Likewise, in order to benefit from the quick release cycles, pay TV operators need to provide quick and clear feedback about the new features being deployed in their environment. The more agile and lightweight operators are in their relationship with their vendors, the more operators are able to reap the benefits of DevOps.

Cisco depends on our customers and partners to provide us with frank and honest feedback about the process and implementation of DevOps. Our chances of improving the process are best when all parties are engaged in providing continuous feedback about what is working and what is not (yet) working.

Way Forward and Summary

At Cisco we are continuously improving the way we implement DevOps. One such way is with cloud deployments, which allow us to more easily deploy systems in an environment that is the same environment that is used in production. As a result, the time from compilation of the code to deployment of the code is getting even shorter. This goes hand in hand with our onward journey to adopt more agile methodologies. We are constantly finding ways to deliver value that meets and exceeds our customer's expectations and to do so in a manner that is fast, reliable, and predictable.

Avraham Poupko has been working with Cisco for more than 20 years. In his current role, he leads a team of customer architects responsible for designing deployments of Cisco® pay TV systems for our valued TV operators. Avraham writes and teaches about topics related to agile methodologies and software architecture.

apoupko@cisco.com

https://il.linkedin.com/in/avrahampoupko

C11-737598-00 07/16