

# A VA Executive's Guide to Development Operations

Office of Technology Strategies (TS), Architecture, Strategy & Design (ASD)  
A VA Executive's Guide to IT Service Management



## Introduction

Is your organization interested in producing reliable information technology (IT) services more rapidly? Development operations or DevOps is a software industry concept that has gained momentum across the public and private sectors to improve IT service delivery to customers. This TS Note explores the cultural shift provided by DevOps towards improving collaboration between the software development, testing, and quality assurance (QA) tasks in order to improve service delivery. It also describes the DevOps alignment to the Agile software development model, with examples of software automation tools that can improve software deployment. It provides an example of how VA has used the continuous delivery technique to improve software development and meet business goals that directly impact Veterans and their families.

## Overview

NDevOps is essential for any organization that aspires to respond rapidly to customer demands when delivering software. DevOps recognizes the interdependence between the programming activities of software development and the IT operational environment that supports software deployment. Even the name of the DevOps concept joins the two workstreams, disallowing them to stand in isolation from each other. How can software development better align to the deployment activities that make software available to customers? How can software deployment be improved?

## The Agile Influence

DevOps is closely associated with the Agile

software development methodology that began in 2001. The IT concept of agility isn't very different than the agility you look for in a player on a basketball court. Agility is mobility, nimbleness, and quickness. To be agile is to effectively respond to change. In software development, agility is required to handle the inevitable changes that occur throughout the project lifecycle.

Ironically, the Agile model emanated from the Ops side of the name - the workstream that is closest to the customer, as shown in the figure below. The Agile model requires that developers work closely with customers to understand their business requirements.

The TS office within OI&T's Architecture, Strategy & Design (ASD) interacts not only with the ASD pillar offices, but also with multiple stakeholders within OI&T and with strategic offices across the enterprise. TS works closely with IT and business owners to capture business rules and provide technical guidance as it relates to Data Sharing across the enterprise, specifically for interagency operability.

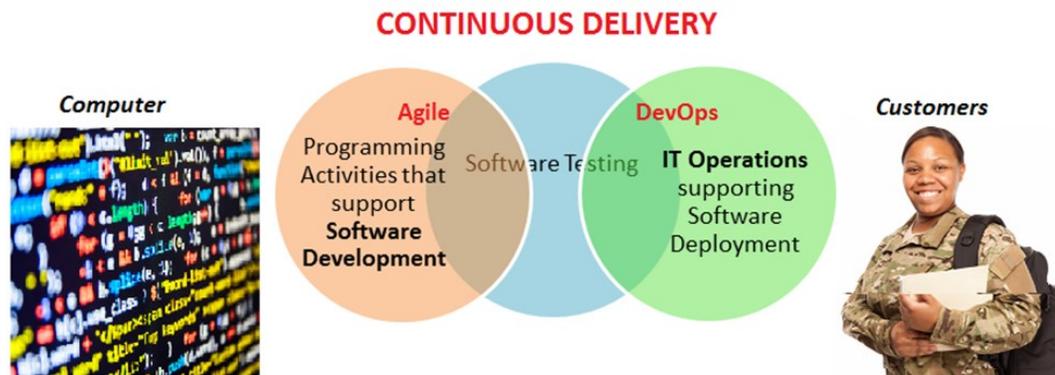


Figure 1 The overlap of tasks for software development and delivery on a continuum from computer code to customers

In 2009, VA began the development of the Veterans Benefits Management System (VBMS), using the Agile approach to automate and digitize claims processing for Veterans. Agile development gained favorable attention for having been instrumental to the success of VBMS for lowering the backlog of claims to the historic low that exists today (about 10% of its peak year backlog). Rather than deliver the software all at once, new features and functionality are added and provided to users at every iteration, with releases scheduled every three months. This gives users the opportunity

to provide valuable feedback to developers, as new features are developed within a continuous development cycle.

## DevOps is Born

It was at an Agile conference in 2008 that developers began to speak more broadly of an "Agile Infrastructure." By the following year, DevOps became the logical continuation of Agile for improving software deployment performance. In highly simplistic terms, you might say that Agile is to software code

# A VA Executive's Guide to Development Operations

Office of Technology Strategies (TS), Architecture, Strategy & Design (ASD)

A VA Executive's Guide to IT Service Management

*development* as DevOps is to software *deployment*. Or put another way, software development is not finished because the code is developed; software development is not finished until the code has been fully tested and is operating effectively.

It was at an Agile conference in 2008 that developers began to speak more broadly of an "Agile Infrastructure." By the following year, DevOps became the logical continuation of Agile for improving software deployment performance. In highly simplistic terms, you might say that Agile is to software code

## Continuous Delivery and Automation Improve Throughput

One way to improve the performance of software deployment is through Agile practices, such as increasing the frequency of the distribution. Firms, such as [Puppet Labs](#), who research and report on the utilization and results of DevOps practices, now have a quantitative measure for IT deployment performance. These include deployment frequency, which describes how frequently an organization deploys code; and deployment lead time, the time required for changes in code to go from "[code committed](#)" to code that is successfully running in the production of the new software release. These are measures of *throughput*, the rate at which software can be developed and deployed. The more frequently software is deployed, the faster the software gets to the customer and the shorter the lead time. High performing developers indicate that their focus on a process of continuous delivery continues after building the code; the same Agile development efforts are powerful tools to enable effective change management as the code flows into production and deployment – this is the DevOps concept. Some start-up firms are now deploying several times each day!

## Software Development from Cradle to Customer

In organizations where there is a separate IT operations department and even a separate software testing function, there is often poor communication between the groups. Therefore, in many organizations, there is no longer the passing of a baton from software development to IT operations that carries an "[I've done my job, now it's your problem](#)" attitude. Rather, instead of development opening up a work ticket and waiting for IT operations to complete testing and other deployment efforts, much of these activities are automated so that developers can accomplish the functions themselves. Software Testing and User Acceptance

Testing have been known to cause [bottlenecks](#) in software development. As a result, there are now "[product-centric](#)" teams that have "[cradle-to-grave](#)" responsibilities.

## DevOps Automation Tools

There are now DevOps tools available to automate the delivery and testing processes used in software development and deployment. [HashiCorp](#) introduced *Atlas* to provide visibility into IT infrastructure, including servers, containers, and virtual machines. [Chef](#) is a systems and cloud infrastructure framework that automates building, deploying, and managing infrastructure by means of short, repeatable scripts called "recipes." It even includes a *Taste Tester*, an automated testing framework. [Puppet Enterprise](#), from Puppet Labs, automates the configuration and management of machines and software. ScriptRock's [GuardRail](#) helps users ensure that their production environment is identical to QA, test, and dev environments. These tools represent just a few of the resources available to ensure a more rapid software deployment.

## Conclusion

The DevOps [principles](#) demand strong communication between the members of the team and a strong understanding of business goals. More than the assistance that software automation tools provide or any special technique, the success of DevOps requires a cultural shift to improve [collaboration](#) between developers and IT operations professionals, including those responsible for QA, testing, and security.

If you have any questions about development operations, don't hesitate to [ask TS](#) for assistance or more information.