Agile Product & Project Management Advisory Service
Executive Update Vol. 12, No. 6

Devops: Beginning with the End in Mind

by Steve Berczuk

One of Steven Covey's *7 Habits of Highly Effective People* is to "begin with the end in mind."[1] When developing software, this is a good rule. Even as teams embrace agile software development and practices that move testing earlier into the development process, the testing of delivery into production is often neglected. Since the goal of software development is to deliver value, and software does not create value until it is in production and available to the the end users, it is worth considering how to bridge the gap between development and operations concerns. Teams can learn from the devops movement and apply practices that can help improve quality and reliability as well as enable more reliable delivery of value and new functionality. This *Executive Update* describes what devops is, its importance in a modern agile organization, and how do learn more about it.

DEVOPS: WHAT, WHY?

Developing software in a way that enables reliable, repeatable deployments should not be a radical idea. Like many named concepts, devops makes formal the practices of many successful teams. This is also true of software development practices for agile and lean, and devops shares some values with agile, especially a focus on delivering value to the business.

The primary benefit of a devops mindset is an improved ability to deliver software that deploys successfully in production the first time. If you have witnessed an application that passed integration testing having a painful deployment in production because of problems relating to configuration and subtle differences between the integration and production environments, you will understand how your organization can benefit from a devops approach.

Your team is producing software to deliver value, and software isn't doing that until it is deployed in a production environment and working reliably. The time and resources spent during deployment, while not part of "development" per se, can be significant. Collaboration between engineering and operations disciplines can bring value to a product in terms of improved reliability and quicker, more effective deployments.

While the value of considering deployment early may be obvious, adding an operations perspective to the development of software requires technical, process, and cultural changes that are similar to those needed to adopt agile.

DEVOPS: AN EXTENSION OF AGILE

While you need not be following agile practices to be interested in closer collaboration between development and operations, devops principles are very similar to those in agile. Agile teams focus on delivering customer value and maintaining code lines that are always working. One goal of developing agile software is a product that is continually deliverable. Agile teams push traditional quality activities, such as testing, closer to the time that the code is developed. That allows for code that builds and runs reliably with every source change, providing clear evidence of the value (or lack of value) that a development effort is delivering.

While an agile team might focus on interaction with customers for requirements validation and frequent testing as well as reviews for validation of functionality, a vital element is missing. Devops extends the loop to include operations team members who will be doing the deployment and testing in the deployment environment. This pushes earlier the concern for the quality of the final deployed product. Much like agile, software development shares the responsibility of testing beyond the testing team and a ultimate testing phase. Devops means that the responsibility for deployment is considered earlier in development.

A devops perspective builds on agile with these principles:

- In waterfall development, testing occurs after delivery and before deployment.
- With agile, testing is considered and executed earlier; everyone cares about build, testing, and architecture, and design includes testability considerations.
- In a devops-oriented environment, you review and test deployment considerations and process at the beginning, as you are designing and developing software. Everyone cares about deployment.
- In the extreme, you can move from doing continuous integration to continuous delivery,[2] where every successful change results in a system that is deployed to the customer. Even if you don't actually deploy each build to a live system by deploying frequently to a "production-like" environment and validating that the end-to-end system works as specified, you can still identify problems earlier and have a more reliable understanding of the state of your project.
- With agile, "done" means a product passes tests and can be demonstrated to a customer. A devops perspective redefines "done" to mean "we know it works in production." While integrating operations into the larger development team is not frequently mentioned in conversations about agile adoption, it's a logical extension. The goal of agile is to more effectively deliver value to stakeholders, and the operations team is a critical part of the value chain.
- To make deployments work more reliably, you can benefit from the whole-team, multidisciplinary approach of agile. Technology decisions should include consideration of operations concerns, rather than solely what makes development easier. To be sure that you are considering these issues, you need to have a representative from the operations team attached to the engineering team, and tests should validate that the software works within operational constraints (and vice versa).
- As with agile teams, successful devops implementations require that people on both the engineering and operations teams become generalizing specialists.[3] Developers will need to understand and know how to work with operations tools and infrastructure, and operations team members may find themselves working on tools for deployment and configuration.

PRINCIPLES AND PRACTICES OF DEVOPS

Devops is multidisciplinary. To deliver and deploy great software reliably, you need collaboration across the whole lifecycle from development to deployment. In practice, this means considering your target deployment environment(s) early and adding processes to make sure your software can be deployed easily, reliably, and consistently in a final system.

This means that you need to collaborate, automate, manage change, and prioritize. Here is each term in more detail:

- **Collaborate** so developers, testers, and operations staff work together to develop processes that will result in the best end-to-end value chain. Taken with automation, this means that operations team requirements must be considered when implementing deployment and configuration tools; for example, the tools should use languages and frameworks with which the operations team can work.
- **Automate** to eliminate manual processes for deployment and reduce the chance of human error, making it easier to identify configuration discrepancies when problems arise, and making it easier (or possible) to practice deploying the software repeatedly, giving your deployment practice the same level of assurance that continual integration gives your code.[4]
- **Manage change** to deployment environments in the same way that you apply to code. All artifacts related to delivering your application should be under a process of version management. This includes operation system, firewall, and machine configurations.
- **Prioritize** means putting deployment and installation processes on the backlog early and getting resources to set up a development resource early. This enables you to practice your deployment, identifying early

problems and architectural issues that add risk to your deployment.

The result of all these practices is that you are able to exercise your deployment to a production environment earlier. Even if you don't fix all the problems you find, you are at least aware of them.

This is a natural extension of integration testing and helps you be more confident in the state of your project. In *Continuous Delivery*, Jez Humble and David Farley express this concisely:

> Crucially, production-readiness also means that the software has had its nonfunctional requirements tested on a production-like environment with a production-sized data set. Any non-functional characteristics that you care about ... should be tested using a realistic load and usage pattern.[5]

ENABLERS

Devops requires some infrastructure and process changes. Take steps toward establishing:

- **Realistic staging environments.** For Web applications, this means such components as servers and network configurations, including firewall. For packaged software, this means target machines in known configurations.
- **Tests for compliance of staging with app expectations.** For example, running a smoke test for the application when you consider changing a firewall rule.
- **An operations team representative as part of the development team.** This ensures that the development team is aware of operation issues and that the operations team is aware of the constraints of the development team.

Considering operations issues early has both capital and time costs, but you can amortize them quickly as you see the savings that result from painless deployments. What most enables devops to be successful has the least direct cost, yet can be the hardest to implement: changing the culture of your engineering and operations teams to be more collaborative.

THE LARGER TEAM

Like agile adoption, while technical practices are useful, the more significant changes involve how people work together. Better communication between operations and engineering teams is a key aspect of the devops approach. Everyone involved in the lifecycle of the application will benefit from moving toward being generalizing specialists, with developers understanding the toolset that makes sense for operations teams, and operations people participating in design and perhaps development tasks related to deployment. Operations personnel are actively developing tools to monitor and deploy software and collaborating with the development team.

This collaboration will quickly lead to everyone understanding possible roadblocks to successful deployments and working to prevent them early when the impact of customer perception and cost will be less.

CONCLUSION

With a devops approach, your development and operations team are collaborating. In a traditional model, the operations team serves the need of the development team, and development is constrained by the requirements of operations.

Devops practices alone cannot find all your problems, but since postdeployment problems are expensive in terms of resources and external appearance, it's worth taking any steps you can to make the last steps of deploying your software be more reliable.

There is nothing magical or new to the idea of devops, and it may seem odd to call out such a natural extension of agile methods. Yet it involves change, and it's worth giving it a name since it's not something all of us do, though perhaps we should.

ENDNOTES

[1]Covey, Stephen R. *The 7 Habits of Highly Effective People*. Free Press, 2004.

[2]Humble, Jez, and David Farley. *Continuous Delivery: Reliable Software Releases Through Build, Test, and Deployment Automation.* Addison-Wesley Professional, 2010.

[3]Ambler, Scott W. "Generalizing Specialists: Improving Your IT Career Skills" (www.agilemodeling.com/essays/generalizingSpecialists.htm).

[4]Humble and Farley. See 2.

[5]Humble and Farley. See 2.

ABOUT THE AUTHOR

Steve Berczuk is an engineer and ScrumMaster at Humedica, where he's helping to build next-generation clinical informatics applications based on software as a service (SaaS). The author of *Software Configuration Management Patterns: Effective Teamwork, Practical Integration*, he is a recognized expert in software configuration management and agile software development. Mr. Berczuk is passionate about helping teams work effectively to produce quality software. He has a master's degree in operations research from Stanford University, a bachelor's degree in electrical engineering from MIT, and is a Certified Practicing ScrumMaster. He can be reached at steve@berczuk.com.