DEVOPS

ITSM AND DEVOPS ARE NOT AT ODDS

By Gene Kim

In thirteen years of studying high-performing IT organizations, I've witnessed the emergence of a new and unsettling trend: whenever I mention ITIL or ITSM in presentations and briefings, people in the audience snicker. When I ask why, they roll their eyes and talk about the shrill, hysterical bureaucrats that suck life out of everyone they touch, doing everything they can to slow the business down by preventing people from getting their work done. But this simply isn't true. In fact, I believe ITSM is more important than ever.

An even more troubling trend is the dismissal of emerging movements as passing fads—movements like DevOps. It's my genuine belief that the patterns and processes of DevOps are the inevitable outcome of applying lean management principles to the IT value stream. DevOps is poised to change IT in a manner we haven't seen since the birth of client-server computing in the 1980s, and the same ITSM practitioners who've been dismissing DevOps are the ones most equipped to support DevOps initiatives and create value for the business.

My goal here is to help you become conversant with DevOps so that you'll be able to recognize DevOps practices when you see them. DevOps complements ITSM, and I hope this article shows you how ITSM practitioners can contribute to this exciting organizational journey.

What Is DevOps?

The term "DevOps" typically refers to the emerging professional movement that advocates a collaborative working relationship between Development and IT Operations, which increases the flow of planned work (i.e., high deploy rates) and the reliability, stability, resilience, and security of the production environment. Why are Development and IT Operations singled out? Because the value stream flows in that direction, from the business (where requirements are defined) to the customer (where value is delivered).

DEVOPS

The DevOps movement began in 2009, at the convergence of a number of adjacent and mutually reinforcing movements, most notably John Allspaw and Paul Hammon's "10 Deploys A Day" and Patrick DeBois's Agile system administration movement. Today, DevOps is more of a philosophical movement; it doesn't have a precise collection of practices—descriptive or prescriptive (e.g., CMM-I, ITIL)—but it does have an incredibly vibrant community of practitioners who are interested in replicating the performance outcomes and cultures described so vividly by organizations like Etsy, Amazon, Netflix, Joyent, and others.

DevOps aims to address a persistent conflict that exists at the core of almost every IT organization, one so powerful it practically preordains horrible outcomes, if not abject failure. The problem? Development is typically measured by feature time to market, which motivates the team to make as many changes as possible, as quickly as possible. IT Operations, on the other hand, is typically measured by uptime and availability. Until very recently, it was impossible to achieve both desired outcomes: fast time to market and sufficient reliability/stability. Because of these diametrically opposed outcomes—make changes very quickly vs. make changes very carefully—Development and IT Operations were in a state of constant battle, with ITSM practitioners in the middle. However, ITIL and ITSM are still the best codifications of the business processes that underpin IT Operations, actually describing many of the capabilities required for IT Operations to support a DevOps-style workstream.

What Are the Basic Principles of DevOps?

In *The Phoenix Project* (2013), Kevin Behr, George Spafford, and I describe the underpinning principles from which all DevOps patterns can be derived as "The Three Ways."

THE FIRST WAY: SYSTEMS THINKING

The First Way emphasizes the performance of the entire system as opposed to the performance of a specific silo. The system can be as large as a division (e.g., Development or IT Operations) or as small as an individual contributor (e.g., a developer, a system administrator), but the focus is on all business value streams that are enabled by IT. In other words, it begins when requirements are identified by the business or IT, built in Development, and then transitioned into IT Operations, where the value is delivered to the customer in the form of a service.

The outcomes of putting The First Way into practice include never passing a known defect to downstream work centers, never allowing local optimization to create global degradation, always seeking to increase flow, and always seeking to achieve a profound understanding of the system (as per Deming).

THE SECOND WAY: AMPLIFY FEEDBACK LOOPS

The Second Way is about creating feedback loops. The goal of almost any process improvement initiative is to shorten and amplify feedback loops so the necessary corrections can be made on a continuous basis. The outcomes of The Second Way include understanding and responding to all customers (internal and external), shortening and amplifying all feedback loops, and embedding knowledge where we need it.

THE THIRD WAY: A CULTURE OF CONTINUAL EXPERIMENTATION AND LEARNING

The Third Way is about creating a culture that fosters two things: continual experimentation, which requires taking risks and learning from success and failure, and the understanding that repetition and practice are the prerequisites of mastery. We need both in equal measure. Experimentation and risk taking are what ensure that we keep pushing to improve, even if it means going deeper into the danger zone than

we've ever gone before; mastery ensures that we can retreat from the danger zone when we've gone too far.

The outcomes of The Third Way include allocating time for the improvement of daily work, creating rituals that reward the team for taking risks, and introducing faults into the system to increase resilience.

The Four Areas of DevOps

AREA 1: EXTEND DEVELOPMENT INTO IT OPERATIONS

In this area, we create or extend the continuous integration and release processes from Development into IT Operations, integrating QA and information security into the workstream, ensuring production readiness, and so forth. The specific steps in this area include:

- Creating a single "repository of truth"
- Creating a one-step build process
- Extending the deployment pipeline processes into production
- Defining roles and integrating QA, information security, and operations/CAB into the Development workstream

First, we put everything we need to rebuild a service into a common repository, including both the application and the environment (i.e., operating system, databases, virtualization, and all associated configuration settings). Then, we make a one-step build process available at the earliest stages of the Development project. By using a common build process and requiring that Development be responsible for ensuring that the code and the environment work together, we'll have an unprecedented level of production ready, even at the earliest stages of the Development project.

This impacts the ITSM process areas of release, change, and configuration management. ITSM practitioners can actively integrate into the DevOps value stream by:

- Finding the automated infrastructure project (e.g., puppet, chef) that provisions servers for deployment. We can help that team by providing release management readiness checklists, security hardening checklists, and so forth, and integrating them into the automated build process.
- Defining preauthorized changes and deployments, and ensuring that production promotions are captured in a trusted system of record that can be reviewed and audited.
- Defining changes and deployments that require authorization, such as security functionalities that are relied upon to secure systems and data (e.g., user authentication modules). The goal is to ensure that changes that could jeopardize the organization (e.g., the infamous 2011 Dropbox failure where customers discovered that authentication was disabled for four hours) never occur.

AREA 2: BUILD IT OPERATIONS FEEDBACK INTO DEVELOPMENT

This area ensures that information from IT Operations is transmitted to Development and the rest of the organization. IT Operations is where value is created, and other departments and divisions need this feedback to make good decisions. The specific steps in this area include:

- Making all infrastructure data visible
- Making all application data visible
- Modifying the incident resolution process and performing blameless postmortems
- Monitoring the health of the deployment pipelines

The first step overlaps with event management, while the second step requires creating a monitoring infrastructure so that there's no excuse for developers not to add telemetry to their applications. The third step then enables IT Operations and Development to re-

solve incidents quickly, by ensuring that all relevant information from the entire application stack is at hand to determine what might have caused the incident and then to restore service. ITSM practitioners can help by ensuring that event, incident, problem, and knowledge management are modified to incorporate Development.

AREA 3: EMBED DEVELOPMENT INTO IT OPERATIONS

According to The Second Way, the goals of this area are to create knowledge and capabilities where they're needed and to shorten and amplify feedback loops. To create tribal knowledge within IT Operations and foster shared accountability for uptime and availability with Development, the steps in this area include:

- Making Development initially responsible for its own services
- Returning problematic services back to Development
- Integrating Development into the incident management processes
- Have Development cross-train IT Operations

AREA 4: EMBED IT OPERATIONS INTO DEVELOPMENT

Area 4 is the reciprocal of Area 3, and the goal of this area is to create the service design and delivery equivalent of designing for manufacturing (DFM). In plant engineering, DFM recognizes that the primary customer of engineering is the manufacturing personnel, and therefore one of engineering's goals is to ensure that parts are designed for easy assembly, minimizing the likelihood of putting parts on backwards, overtightening parts, damaging parts during transit or assembly, and so forth.

In a similar fashion, in addition to ensuring that the needs of IT Operations are integrated into Development's daily processes of design, requirements specification, development, and testing, the product and

processes are designed with resiliency in mind. The steps in this area include:

- Embedding IT Operations knowledge and capabilities into Development
- Designing for IT Operations
- Institutionalizing IT Operations knowledge
- Breaking things early and often

This includes embedding IT Operations resources into Development, creating reusable user stories for the IT Operations staff (i.e., deployment, management of the code in production, etc.), and defining nonfunctional requirements that can be used across all projects.

ITSM and the DevOps movement are not at odds. Quite the contrary—they're a perfect cultural match. As DevOps gains momentum, I'll be excited to see what we can achieve with this winning combination. I hope you now have a better understanding of what DevOps is and why it's so important. Think of the possibilities! And when you're done thinking, put those ideas and practices into place in the IT organizations you support.

About the Author

Gene Kim, founder and former CTO of Tripwire, is an award-winning CTO, researcher, and author. He's written three books, including ***The Visible Ops Handbook*** and ***The Phoenix Project: A Novel About IT, DevOps, and Helping Your Business Win***. He's worked with some of the top Internet companies on improving deployment flow and increasing the rigor of operational processes. In 2007, Computerworld added Gene to the "40 Innovative IT People Under the Age of 40" list, and Purdue University's Department of Computer Sciences recognized him with Outstanding Alumnus Award for achievement and leadership in the profession.

Top 10 Things You Need to Know About DevOps

1 What Is DevOps and Where Did It Come From?

The term "DevOps" typically refers to the emerging professional movement that advocates a collaborative working relationship between Development and IT Operations, resulting in the fast flow of planned work (i.e., high deploy rates) while simultaneously increasing the reliability, stability, resilience, and security of the production environment.

Why Development and IT Operations? Because that's typically the value stream that exists between the business (where requirements are defined) and the customer (where value is delivered).

The origins of the DevOps movement are commonly placed around 2009, at the **convergence** of numerous adjacent and mutually reinforcing movements:

- The Velocity Conference movement, especially the seminal **"10 Deploys A Day"** presentation given by John Allspaw and Paul Hammond
- The "infrastructure as code" movement (Mark Burgess and Luke Kanies), the Agile infrastructure movement (Andrew Shafer) and the Agile system administration movement (Patrick DeBois)
- The "Lean Startup" movement by Eric Ries
- The continuous integration and release movement by Jez Humble
- The widespread availability of cloud and PaaS technologies (e.g., Amazon Web Services).

2 How Does DevOps Differ from Agile?

One tenet of the Agile development process is to deliver working software in smaller and more frequent increments, as opposed to the the "big bang" approach of the waterfall method. This is most evident in the Agile goal of having potentially shippable features at the end of each sprint (typically every two weeks).

High deployment rates will often pile up in front of IT Operations for deployment. According to Clyde Logue, founder of StreamStep, "Agile was instrumental in Development regaining the trust in the business, but it unintentionally left IT Operations behind. DevOps is a way for the business to regain trust in the entire IT organization as a whole."

DevOps is especially complementary to the Agile software development process, as it extends and completes the continuous integration and release process by ensuring the code is production-ready and providing value to the customer.

DevOps enables a far more continuous flow of work into IT Operations. When code isn't promoted into production as it's developed (e.g., Development delivers code every two weeks, but it's deployed only every two months), deployments will pile up in front of IT Operations, customers won't get value, and the deployments will most often result in chaos and disruption.

DevOps has an **inherent cultural change component**, as it modifies the the flow of work and local measurements of Development and IT Operations.

3 How Does DevOps Differ from ITIL or ITSM?

Although many people view DevOps as backlash to ITIL or ITSM, I take a different view. ITIL and ITSM are still the best codifications of the business processes that underpin IT Operations, and they actually describe many of the capabilities needed in order for IT Operations to support a

DevOps-style work stream.

Agile and continuous integration and release are the outputs of Development, which are the inputs into IT Operations. In order to accommodate the faster release cadence associated with DevOps, many areas of the ITIL processes require automation, specifically around the change, configuration, and release processes.

The goal of DevOps is not just to increase the rate of change, but to successfully deploy features into production without causing chaos and disrupting other services, while quickly detecting and correcting incidents when they occur. This brings in the ITIL disciplines of service design, incident, and problem management.

4 How Does DevOps Fit with Visible Ops?

Visible Ops is a prescriptive guide to capture the "good to great" transformations of the high performing IT Operations, and one of the key concepts is the notion of controlling and reducing unplanned work.

In many ways, I view DevOps as the inverse, focusing primarily on how to create a fast and stable flow of planned work through Development and IT Operations. However, DevOps also has a more holistic approach to the systematic eradication of unplanned work, addressing principles of resilient engineering and the responsible management and reduction of technical debt.

5 What Are the Underpinning Principles and Patterns of DevOps?

In the **DevOps Cookbook** and **When IT Fails: A Business Novel**, my fellow authors and I describe the underpinning principles from which all DevOps patterns can be derived as "The Three Ways." In the same way, the patterns of DevOps are divided into **four areas**. You read a summary of "The Three Ways" and the four areas in the preceding article.

6

What Is the Value of DevOps?

I believe there are three business benefits that organizations get from adopting DevOps:

- Faster time to market (i.e., reduced cycle times and higher deploy rates)
- Increased quality (e.g., increased availability, increased change success rate, fewer failures)
- Increased organizational effectiveness (e.g., increased time spent on value adding activities vs. waste, increased amount of value being delivered to the customer)

FASTER TIME TO MARKET

In 2007, the IT Process Institute **benchmarked** over 1,500 IT organizations and concluded that high-performing IT organizations were on average five to seven times more productive than their non-high-performing peers. They were making fourteen times more changes, with one-half the change failure rate, four times higher first-fix rates, and ten times shorter Severity 1 outages. They had four times fewer repeat audit findings; they were five times more likely to detect breaches by an automated internal control; and they had eight times better project due date performance!

In the research, the highest deploy rate observed was approximately 1,000 production changes per week, with a change success rate of 99.5 percent. We thought this was fast...

One of the characteristics of high performers is that they accelerate away from the herd. In other words, the best continue to get even better. This is absolutely happening in area of deploy rates. Organizations that are employing DevOps practices are outperforming the highest performers by orders of magnitude. Amazon, for instance, has gone on record stating that they're doing over 1,000 deploys a day, sustaining a change success rate of 99.999%!

The capability to sustain high deploy rates (i.e., fast

cycle times) translates into business value in two primary ways: how quickly the organization can go from an idea to value being delivered to the customer ("concept to kaching," as Damon Edwards and John Willis say), and how many experiments the organization can be performing simultaneously.

There is no doubt in my mind that if I'm in an organization where I can only do one deployment every nine months, and my competitor can do ten deployments in a day, I have a significant, structural competitive disadvantage.

High deploy rates also enable rapid and constant experimentation. Scott Cook, the founder of Intuit, has been one of the most outspoken advocates for a "rampant innovation culture" at all levels of the organization. According to Cook, "every employee [should be able to] do rapid, high-velocity experiments...Dan Maurer runs our consumer division, including running the TurboTax website. When he took over, we did about seven experiments a year. By installing a rampant innovation culture, they now do 165 experiments in the three months of tax season. Business result? Conversion rate of the website is up 50 percent. Employee result? The folks just love it, because now their ideas can make it to market."

To me, the most shocking part of Scott Cook's story is that they were doing all these experiments during peak tax-filing season! Most organizations have change freezes during their peak seasons (e.g., retailer may often have holiday change freezes from October until January), but if you can increase conversion rates, and therefore sales, during peak seasons when your competitor cannot, then that's a genuine competitive advantage.

The prerequisites to do doing this include being able to do many small changes quickly, without disrupting service to customers.

REDUCED AMOUNT OF IT WASTE

Mike Orzen and I have long talked about the enormous **waste in the IT value stream**, caused by long lead times, poor hand-offs, unplanned work, and re-

work. How much value could be recaptured by applying DevOps-like principles? We calculated that if we could halve the amount of IT waste and redeploy those dollars in a way that could return five times what was invested, we would generate $3 trillion dollars of value per year.

That's a staggering amount of value and opportunity that we're letting slip through our fingers. That's 4.7 percent of our annual global GDP—and more than the entire economic output of Germany. The potential economic impact on productivity, standards of living, and prosperity almost makes this a moral imperative.

However, there's an even greater cost. Working in most IT organizations is often thankless and frustrating. People feel trapped, helpless to change the outcome. Management abdicates their responsibility to ensure that IT is managed well, resulting in endless intertribal warfare between Development, IT Operations, and information security. And things only get worse when the auditors show up.

What inevitably results is chronic underachievement. The life of an IT professional is often demoralizing and frustrating. It typically leads to feelings of powerlessness and stress that seep into every aspect of life. From stress-related health problems, to social issues, to tension at home, being an IT professional is not only unhealthy, but likely unsustainable.

7 How Do Information Security and Quality Assurance Integrate into a DevOps Workstream?

The high deployment rates typically associated with DevOps workstreams will often put enormous pressure on quality assurance and information security. Consider the case where Development is doing ten deployments per day, while information security requires a four-month lead time to turn around application security reviews. At first glance, there appears to be a fundamental mismatch between the rate of code development and code testing.

An example of the risk posed by insufficiently tested deployments is the famous **2011 Dropbox failure**, where authentication was turned off for four hours, during which unauthorized users had access to all stored data.

The good news for quality assurance and information security is that Development organizations capable of sustaining high deployment rates are likely using continual integration and release practices, which often go hand in hand with a culture of continuous testing. In other words, whenever code is checked in, automated tests are automatically run, and issues must be fixed right away, just as if a developer checked in code that didn't compile.

By integrating functional, integration, and information security testing into the daily operations of Development, defects can be found and fixed more quickly than ever. (There are many information security tools, such as Gauntlet and Security Monkey, that help test security objectives in the development and in production processes.)

A genuine concern is that static code analysis tools take too long to run to integrate into a continuous integration and testing process, often requiring hours or even days to complete. In these cases, information security should designate the specific modules that security functionality relies upon (e.g., encryption, authentication modules). Whenever those modules change, a full retest should be run; otherwise, deployments can proceed.

One last note: We've observed that DevOps workstreams often rely more on detection and recovery than standard functional unit testing. In other words, when doing development for packaged software, where it's very difficult to deploy changes and patches, quality assurance relies heavily on testing the code for functional correctness before it is shipped. On the other hand, when software is delivered as a service and defects can be fixed very quickly, then quality assurance can reduce its reliance on testing, and instead rely more on production monitoring to detect defects in production, as long as they can be quickly fixed.

Quick recovery from code failures can be aided by using "feature flags," which enable and disable code functionality via configuration settings, instead of having to roll out entirely new deployments.

Relying on detection and recovery for quality assurance is obviously far more applicable when the worst that could happen is the loss of functionality or required performance. However, when failures risk the loss of confidentiality or integrity of systems or data, then reliance cannot be put on detection and recovery—instead, it must be tested before code is deployed, because a production failure would generate a genuine security incident.

8 My Favorite DevOps Pattern

All too often in software development projects, Development will use up all the time in the schedule on feature development. This leaves insufficient time to adequately address IT Operations issues. Shortcuts are then taken in defining, creating, and testing everything the code relies upon, which includes the databases, operating systems, network, virtualization, and so forth.

This is certainly one primary cause of the perpetual tension between Development and IT Operations, as well as the suboptimal outcomes. The consequences of this are well known: inadequately defined and specified environments, no repeatable procedures to deploy them, incompatibilities between deployed code and the environment, and so on.

When Development is using an Agile process, we can do something very simple and elegant: According to Agile, we're supposed to have working, shippable code at the end of each sprint interval. We can modify the Agile sprint policy so that instead of having shippable, viable code at the end of each sprint, you also have to have the environment that it deploys into (at the earliest sprint, so we're talking sprint 0 and sprint 1).

Instead of having IT Operations responsible for creating the specifications of the production environment, in-

stead, they will build an automated environment creation process. This mechanism will create the production environment, but also the environments for Development and quality assurance.

By making environments (and the tools that create them) available early, perhaps even before the software project begins, developers and quality assurance can run and test their code in consistent and stable environments, with controlled variance from the production environment.

Furthermore, by keeping variance between the different stages (e.g, development, quality assurance, integration test, production) as small as possible, we can find and fix interoperability issues between the code and environment long before production deployment.

Ideally, the deployment mechanism you build will be completely automated, and there are many tools for that, including shell scripts, Puppet, Chef, Solaris Jumpstart, Redhat Kickstart, Debian Preseed, and more.

9 My Second-Favorite DevOps Pattern

One of my favorite quotes is from Patrick Lightbody, former CEO of BrowserMob, who said, "We found that when we woke developers up at 2 a.m., it was a phenomenal feedback loop. Defects got fixed faster than ever."

This underscores the problem where Development checks in their code at 5 p.m. on Friday, high-fives each other in the parking lot, and then goes home, leaving IT Operations to clean up the mess the entire weekend. Worse, defects and known errors keep recurring in production, forcing IT Operations to continually firefight, and the root cause is never fixed because Development is focused on building new features.

An important element of the Second Way is to shorten and amplify feedback loops, and to bring Development closer to the customer experience (which includes IT Operations and the end users of the service being delivered).

Note the symmetry here: My favorite pattern is all about embedding IT Operations into Development; my second-favorite pattern is about putting Development into IT Operations.

Here, we put Development into the IT Operations escalation chain, possibly putting them in level 3 support, or even having Development be completely responsible for the success of the code deployments, either rolling back or fixing forward until service is restored to the customer.

The goal is not to have IT Operations replaced by Development. Instead, it's to ensure that Development sees the downstream effects of their work and changes, and has walked in the shoes of IT Operations enough to be motivated to fix issues quickly.

10 My Third-Favorite DevOps Pattern

Another recurring problem that occurs in the DevOps valuestream between Development and IT Operations isn't sufficiently standardized. For example, when every deployment is done differently, every production environment is a special snowflake. When this occurs, no mastery is ever built in the organization in terms of either procedure or configuration.

In my third-favorite pattern, the organization defines reusable deployment procedures that can be used across projects. There's a very elegant solution in the Agile methodology for this, where deployment activities are turned into user stories. For example, you could build a reusable user story for IT Operations called "Deploy into high availability environment," which defines the steps to build the environment, as well as how long it will take, the resources that will be required, etc.

This information can then be used be used by project managers to accurately integrate the deployment activities into the project plan. For instance, we would have high confidence in the deployment schedule if we knew that the "Deploy into high availability environment" sto-

ry has been executed fifteen times in the past year, taking an average of three days, plus or minus one day. Furthermore, we also gain confidence that the deployment activities are being properly integrated into every software project.

Recognizing that certain software projects require unique environments that IT Operations doesn't officially support, we can allow for exceptions where these environments are allowed in production, but are supported by someone outside of IT Operations (i.e., unsupported environments).

By doing this, we get the benefits of environment standardization (e.g., reduced production variance, fewer snowflakes in production, increased ability for IT Operations to reliably support and maintain) while allowing for special cases that allow for the nimbleness businesses sometimes require.

The Top Ten

1. The Origins of DevOps
2. DevOps vs. Agile
3. DevOps vs. ITIL/ITSM
4. DevOps and Visible Ops
5. The Principles and Patterns of DevOps
6. The Value of DevOps
7. DevOps, Quality Assurance, and Information Security
8. Gene's Favorite DevOps Pattern
9. Gene's Second-Favorite DevOps Pattern
10. Gene's Third-Favorite DevOps Pattern