DEVOPS COOKBOOK FOR ERP TEAMS

How ERP teams can enjoy the advantages of DevOps without incurring the risk of noncompliance

By Dennis Nelson, Solution Architect, Quest Software

Quest®

ERP teams — on the outside of DevOps, looking in

"Our Java, Ruby and .NET application teams have been working in DevOps for years. Why can't our ERP teams do DevOps?"

This is a fair question, and more ERP teams are asking it.

Like all teams, they want to benefit from the advantages of DevOps, like shorter cycle times and less unplanned work (break-fix). They want to introduce tools to their application lifecycle to automate as much of the process as possible. They want to reduce the effort — and margin of human error — in moving code from development to test to production. Most of all, they want to meet deadlines for delivering the changes and improvements the business demands without breaking any part of the system.

IF ALL THE COOL KIDS HAVE DEVOPS, THEN WHAT'S KEEPING THE ERP TEAMS FROM GETTING IT?

In short: risk.

ERP teams face a different environment than other application teams face because they work with the organization's financial system of record. Progress toward DevOps can introduce risk of noncompliance for which most people — auditors, change control advisors, compliance managers, even life-long ERP developers — have little appetite.

This eBook explores the potential role of DevOps on ERP teams. It offers a framework for guiding ERP teams toward a DevOps culture, with workflows that can speed up deployment while still mitigating risk.

ERP teams will see that, in some ways, they may already be doing DevOps and not know it. They can enjoy the advantages of DevOps without incurring the risk of noncompliance.

What's the difference between DevOps and agile?

This is another fair question. Until a company adopts one or both of them and people see how they work and what they affect, the two concepts are often conflated, but they are not interchangeable.

THE WAY THEY REGARD PEOPLE AND PROCESS

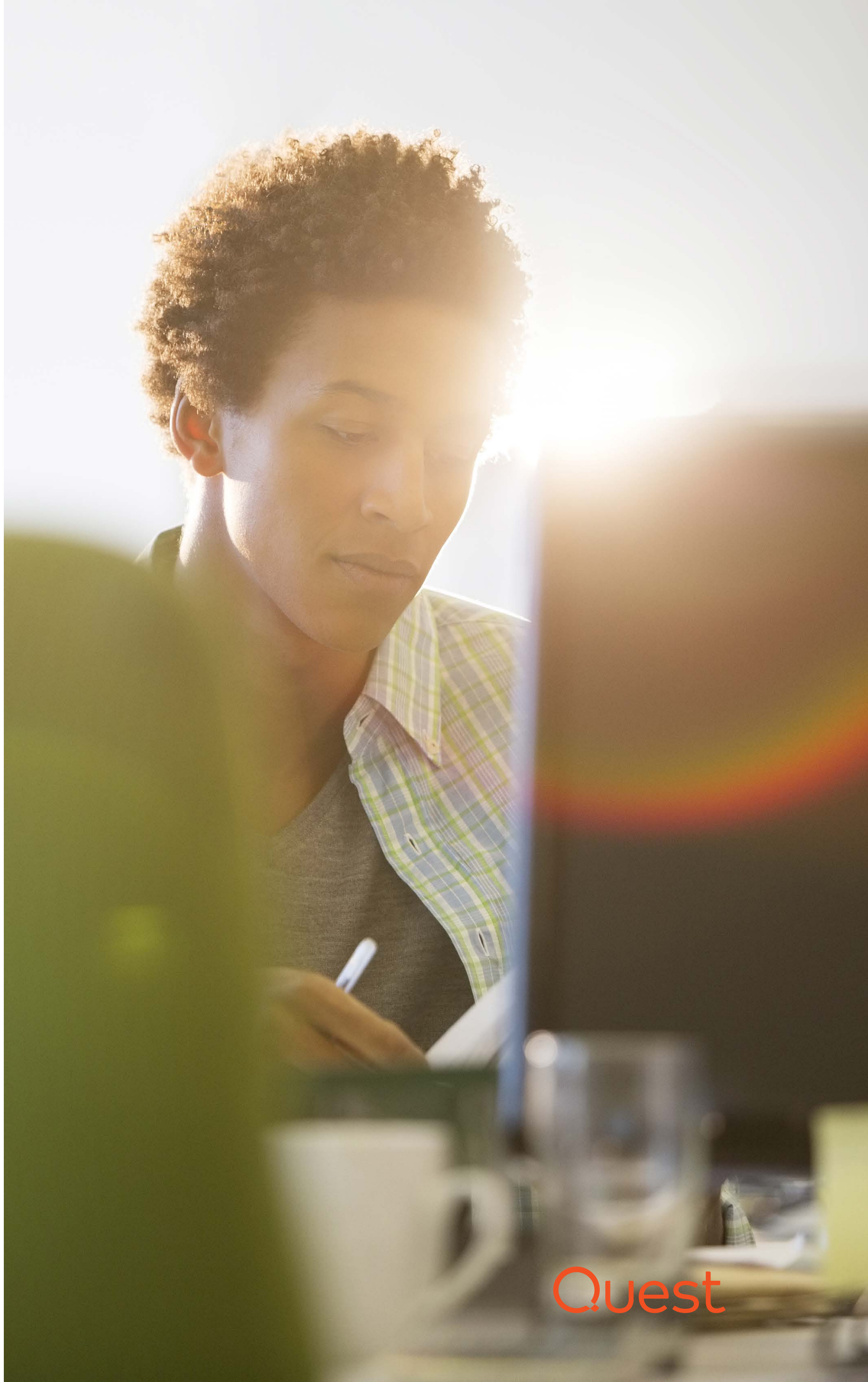
In the context of ERP development, one important aspect of agile is that it "value[s] individuals and interactions over processes and tools."[1] On the face of it, agile is people-dependent rather than process-dependent, which can lead to the bottlenecks that DevOps is trying to fix. People who are trying to do too much become the bottleneck because everybody has only so much bandwidth.

That often happens in ERP environments. Everything piles up on the plate of the one technical lead who has the most knowledge of the system. When that lead is unable or unavailable to work on a problem, delays and bottlenecks ensue. Some agile processes depend on each team member being able to perform every job on the team because it increases understanding and makes close communication much easier. That way, nobody is waiting for a specialist to become available for the next step in the process.[2]

DevOps, however, assumes separation between development teams (Dev) and IT operations teams (Ops), with frequent, regular communication, but without members moving between teams to perform one another's duties.

1 Beck, Kent, et al., "Manifesto for Agile Software Development," 2001, http://agilemanifesto.org/

2 Franklin Jr., Curtis "Agile Vs. DevOps: 10 Ways They're Different," Information Week, July 5, 2016, https://www.informationweek.com/devops/agile-vs-devops-10-ways-theyre-different/d/d-id/1326121



The point of DevOps is to deliver stable, nondisruptive technology that keeps pace with the needs of the business. That involves defining repeatable processes and automating as many of them as possible. Knowledge is shared across team members not so that people can replace one another but so that they can still keep work flowing even when technical leads are not available. With better-defined processes that include other people who all know what needs to be done, no single person can be a bottleneck in the development process.

THE WAY THEY FOCUS ON AUTOMATION

Another priority of the agile manifesto is "responding to change over following a plan," which is often related to rapid deployment and frequent releases. Greater frequency is one aspect of the DevOps goal, but it is not the paramount consideration.

In DevOps, greater frequency and reliability are related to automated deployment, which depends on certain tools for functions like versioning, testing and releasing software. Agile development is less dependent on automation. Teams can use tools, but their choice of tool is not as integral to the goals of agile.

THEIR TOLERANCE FOR UNPLANNED WORK

Repeatable processes and automation are at the heart of what DevOps seeks to accomplish, but they can get lost by ERP teams plunging ahead into agile development. They may adopt tools like Jira, Bitbucket and Jenkins to introduce more automation, which may lead to some improvement. But if they keep their poorly defined processes, it's still not DevOps.

The goals are to reduce cycle time — the time between requesting a change and getting it into production — and to reduce unplanned work, also known as break-fixing. Automation tools can help move more bytes into production faster, but if that means going back and using yet more development cycles to fix the work, then releases will never be truly stable.

Unplanned work and break-fixes take away the business value of automation. In the DevOps world, they should be limited or eliminated. They affect schedules and cause application users to lose confidence in the ability of the DevOps team to deliver working code on time.

The Three Ways of DevOps

Once DevOps is disentangled from agile, ERP developers can more easily gauge its fit with their own teams. A useful approach is the Three Ways of DevOps.[3]

THE FIRST WAY — THINK ABOUT THE ENTIRE SYSTEM

The First Way focuses on left-to-right workflow and seeks to create a delivery system that can release reliable software to the customer or user. DevOps breaks down the barriers, stove-piped processes and perspectives of developers, administrators, QA and business stakeholders. The goal is for ERP developers, for example, to understand how their code may affect others as it moves from left to right through the organization.

DevOps is about having greater insight and transparency through the entire delivery system. This means that all team members can see a request from the time a business manager submits it until the time it is implemented and the customer realizes the envisioned benefits.

Practices include:

- Continuous build, integration and deployment
- Creating environment on demand
- Limiting work in process
- Building safe systems and organizations that are safe to change

3 This section based on Behr, Kevin; Spafford, George and Kim, Gene, "The Phoenix Project: A Novel About IT, DevOps, and Helping Your Business Win," 2013, https://itrevolution.com/book/the-phoenix-project/

THE SECOND WAY — AMPLIFY FEEDBACK LOOPS

The Second Way emphasizes the constant, right-to-left flow of feedback from later stages of development to earlier ones. The objective is to identify problems as quickly as possible and either prevent their recurrence or enable faster detection and recovery. That happens by continually getting knowledge back to the source, where it needs to be.

Taken to its logical conclusion, that means not deploying defective code or builds. The Second Way involves stopping deployment and fixing defects before proceeding. The opposite is to deploy defects and fix them in the next environment downstream. For example, Dev should never throw code or test defects over the wall to Ops for them to figure out and fix.

Practices in the Second Way include:

- "Stopping the production line" when builds and tests fail
- Constantly elevating the improvement of daily work over the daily work itself
- Creating fast, automated test suites to ensure that code is always in a deployable state
- Creating shared goals and shared pain between Development and IT Operations
- Creating pervasive production telemetry so everyone can see whether code and environments are operating as designed and customer goals are met

THE THIRD WAY — EXPERIMENT AND LEARN

The goal of The Third Way is to create a culture that nurtures continual experimentation and an understanding that repetition and practice are the prerequisite to mastery. Experimentation involves taking risks and learning from both success and failure to improve the system of work.

In the Third Way, necessary practices include:

- Creating a culture of innovation and risk taking
- Creating a culture of high trust
- Allocating at least 20 percent of development and IT operation cycles towards nonfunctional requirements
- Constant reinforcement that improvements are encouraged and celebrated

Creating a culture of innovation and risk taking is a stretch for most ERP teams. They are usually too preoccupied with user requests, required patches, trouble tickets and upgrades to think about innovation. Besides, they are paid to help avoid risk, not look for it.

And that gets to the main difference between ERP development and other development efforts in the organization.

Why is ERP different?

Some teams can appoint a scrum manager and a release manager, then implement Jenkins, Git and Jira and consider themselves on the road to DevOps. That approach doesn't usually fit an ERP team, for a couple of reasons.

CONTINUOUS INTEGRATION/CONTINUOUS DEPLOYMENT (CI/CD)

First of all, CI/CD is certainly different.

In a Java or .NET application, for example, the CI process, in its most basic form, polls the version control system to see if any commits have occurred. If so, it checks out the latest version of the software, runs the build script to compile the software, runs the tests and notifies stakeholders of the results. That makes sense for applications that are compiled and linked to libraries and configuration files. The main goal of the process is to ensure that no code checked in to the version control system breaks the build.

ERP systems like Oracle E-Business Suite (EBS) and PeopleSoft are different. On one side, the vendor, Oracle, provides upgrades and updates to core functions. On the other side, ERP teams modify their customizations, extensions, interfaces, reports, workflows and any other object types unique to their implementation of the ERP system. There is no complete rebuild per se of the ERP system every time a commit happens.

ERP teams need to think of CI/CD as it relates to their customizations, extensions, interfaces, reports, workflows and other object types. For example, in the case of an Oracle EBS report, the build should include all associated objects such as the .rdf file, concurrent program, concurrent executable, value sets, views, package specification and package body. All objects associated with a report should be part of the build, testing and deployment process.

COMPLIANCE. OR ELSE.

The prevailing theme of ERP is compliance. For example, whatever goes on in the financial system of record must be demonstrably compliant with laws like Sarbanes-Oxley (SOX). CRM systems have requirements for personally identifiable information (PII) and the General Data Protection Regulation (GDPR). Health Care Management (HCM) systems must comply with the Health Insurance Portability and Accountability Act (HIPPA). The standard of accountability is usually more rigorous in ERP systems than for application development or even database development.

Meanwhile, the prevailing theme of DevOps is moving forward. People are informed about the status of each release and each object in a release. Nothing is hidden and everybody knows exactly what's already built and ready to go. If a defect makes it into production, it's still possible to roll back just the bits that don't work and get to a stable state, instead of having to roll back the entire release. The whole point of DevOps is to keep moving to the right and not go back.

So, if it's the ERP team's job to keep audit and compliance in mind — especially if the company is publicly traded — how can it do that while adopting DevOps?

For one thing, it is possible to maintain appropriate controls if the right kind of tool manages the entire process. For another, the DevOps team must include the auditors and compliance managers themselves in the process.

WHY? SO WE CAN AGREE ON ACCEPTABLE LEVELS OF RISK.

Even though they are notoriously risk-averse, stakeholders like compliance managers and auditors must be part of the DevOps conversation. Standing still and changing nothing appears to incur the least risk, but it is not an option, for several reasons:

- Business users and customers want changes and improvements, and they don't want to wait forever for them.
- Aversion to the changes that keep business systems up to date can actually introduce risk, with secure information and applications being made vulnerable to breach.
- If management intended to avoid risk completely, it probably would not hire ERP developers in the first place.

Consider a company subject to SOX reporting requirements. It wants no high-priority findings published in its SEC filings. Therefore, it probably has little appetite for the risk that may be introduced when making changes to its ERP application change management process.

Given the need to move things forward to respond to the needs of business, what kind of balance can be struck with regard to risk?

A move to DevOps in that organization would have to begin with an agreement between ERP developers and compliance managers about the level of risk the business is willing to accept. The developers would move things through their process, using automation when possible, but the compliance managers would not need to give up their strong controls. Their agreement would mitigate risk down to a quantifiable target; for example, no high-priority findings on a SOX audit report.

That conversation is quite unlike the one the application developers and database developers have when they adopt DevOps.

That is why ERP is different.

Why should ERP teams adopt DevOps?

A few salient metrics illustrate the benefits that high-performing IT organizations enjoy by having adopted DevOps practices[4]:

GREATER THROUGHPUT AND STABILITY

- High performers deploy code hundreds or thousands of times per day, compared to once per month or every few months for low performers.
- The lead time to go from code committed to code deployed and running successfully ranges from one hour for high performers to several months for low performers.
- The percentage of changes resulting in outage is 0 to 15 for high performers and 16 to 30 for low performers.

UNPLANNED WORK AND BREAK-FIXING

- New work occupies 49 percent of workers' time in high-performing companies and 38 percent in low-performing companies.
- Unplanned work, such as break-fixing, dealing with emergencies and answering audit requests, occupies only 21 percent of time for high performers and as much as 27 percent for low performers.

SECURITY

- When developers build security into their daily work rather than adding it in later, they can spend about 50 percent less time addressing security problems.

The fact is that DevOps and ERP are not mutually exclusive. Many of the techniques and practices inherent to DevOps — simplifying server management with automation, improving versioning for package migration, creating reusable frameworks, automating and grouping tasks — also apply to ERP development.[5]

4 From "2016 State of DevOps Report," Puppet, Inc., and DevOps Research and Assessment," June 2016, https://puppet.com/resources/whitepaper/2016-state-of-devops-report

5 Pruess, Rebecca, "DevOps for Oracle Apps: 10 E-Business Suite Development Tips," DevOps Zone, February 12, 2019, https://dzone.com/articles/devops-for-oracle-apps-10-e-business-suite-develop

Are some ERP teams doing DevOps without even knowing it?

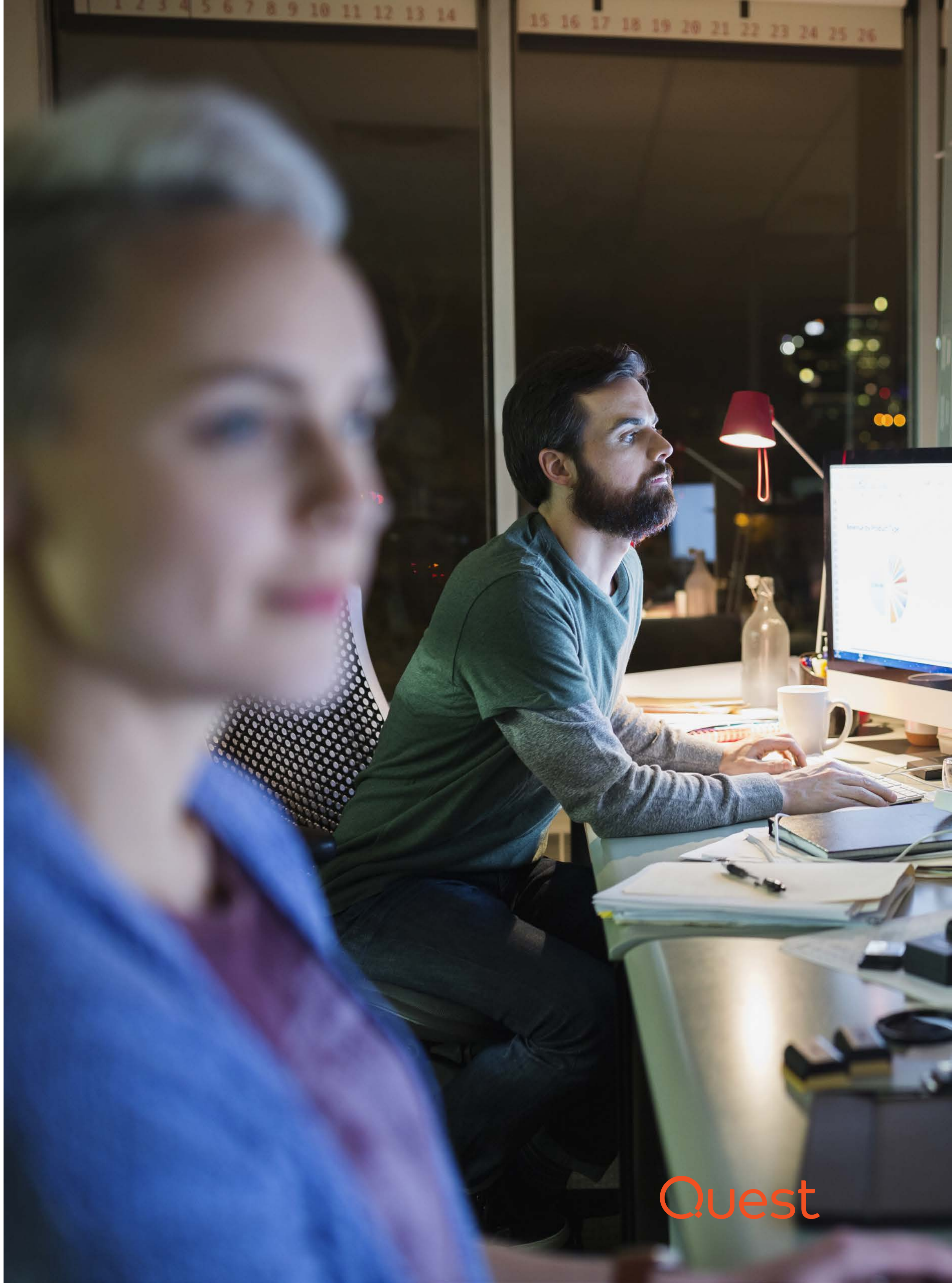
"YOU CAN'T REALLY DO DEVOPS IN AN ERP ENVIRONMENT."

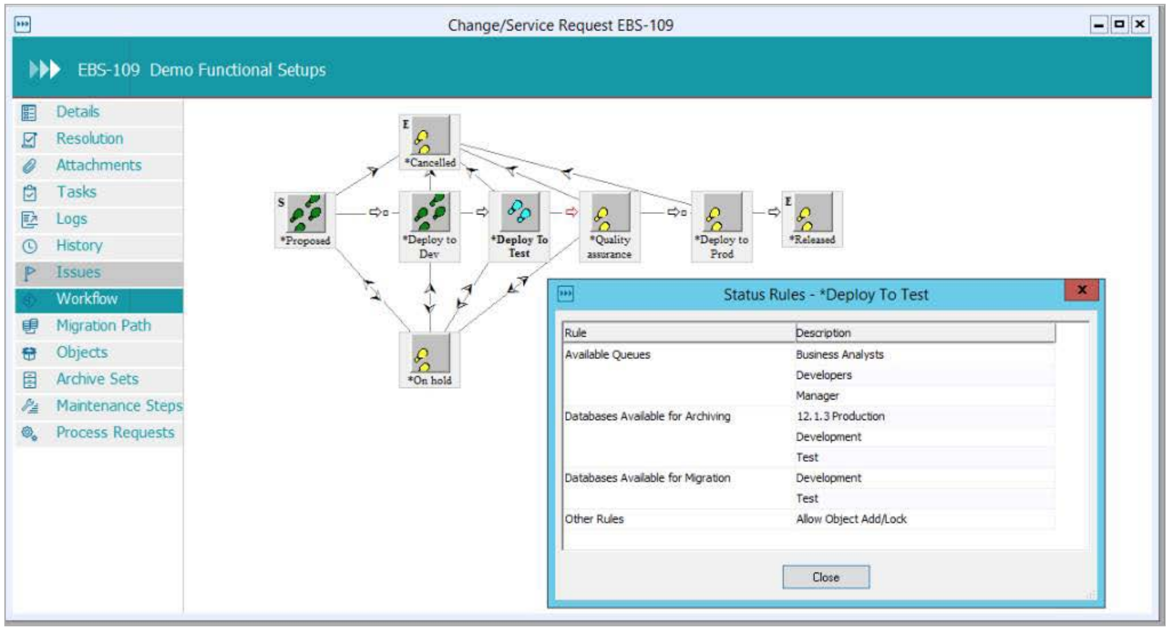
For the reasons cited above, many developers think that DevOps and ERP do not mix. Not just Java developers doing departmental-type applications, but developers who have built a whole career on products like PeopleSoft and Oracle E-Business Suite.

Not only are DevOps and ERP not mutually exclusive, but practices and preventative controls common to DevOps have long been present in tools commonly used by ERP developers; notably, Stat® from Quest.

As noted, the audit committee and compliance managers set the risk appetite for the business. They define the general control objectives and what needs to be in place to adhere to them. If ERP development strays from those objectives because developers have chosen a tool that doesn't support them, the organization could end up with high-priority audit findings.

That makes everybody think that you can't really do DevOps in an ERP environment.





Rule	Description
Available Queues	Business Analysts
	Developers
	Manager
Databases Available for Archiving	12.1.3 Production
	Development
	Test
Databases Available for Migration	Development
	Test
Other Rules	Allow Object Add/Lock

Figure 1: Stat workflow and status rules

YES, YOU CAN. IF YOUR TOOL SUPPORTS PREVENTATIVE CONTROLS AND THE THREE WAYS.

Stat enforces preventative controls in the ERP system that keep the organization within its risk-comfort zone. DevOps automation tools like Jira and Jenkins can find defects and missing approvals, but they focus downstream from the source of the problem, after deployment or migration. By then, the mistakes have already been made.

Furthermore, Stat delivers many of the functions needed to bring ERP development in line with the most important DevOps principles, as outlined in The Three Ways.

Stat and the First Way

Stat supports **left-to-right workflow** by allowing the organization to define processes as business rules, transfer rules, status rules and approval requirements. As shown in Figure 1, it also offers stakeholders a quick view into the status of change requests, the progress that has been made so far and the steps yet to be completed.

Next, the Stat features of archive set, release management and mass migration functionality correspond to **continuous build, continuous integration and continuous deployment**.

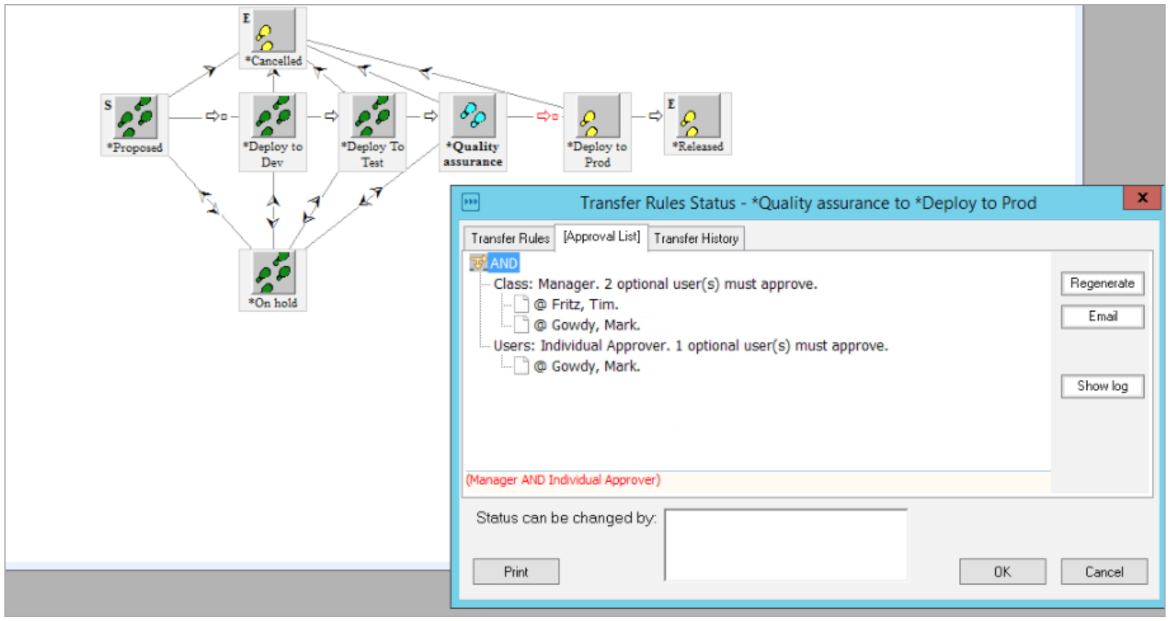
Those are the DevOps concepts behind the nonstop release of reliable software to customers and users.

Finally, to **create environment on demand**, Stat offers Environment Refresh change service requests (CSRs) and environment refresh tracking. It allows IT teams to sync changes from one environment to a completely new environment.

Stat and the Second Way

Supporting the DevOps goal of **fixing defects before they move downstream**, Stat's status and transfer rules can be configured to prevent builds from being deployed. Stat makes it possible to capture issues and track them all the way to resolution. It enables ERP developers to continuously improve their coding practices by offering insight into root-causes and corresponding fixes.

Workflow in Stat provides for preventative controls to **enforce change management policies**. Stat allows deployment of changes in the ERP system only when they meet preconfigured business rules that reflect those policies, such as approval by particular business managers, sign-off for user acceptance testing or review by the change control advisory board.

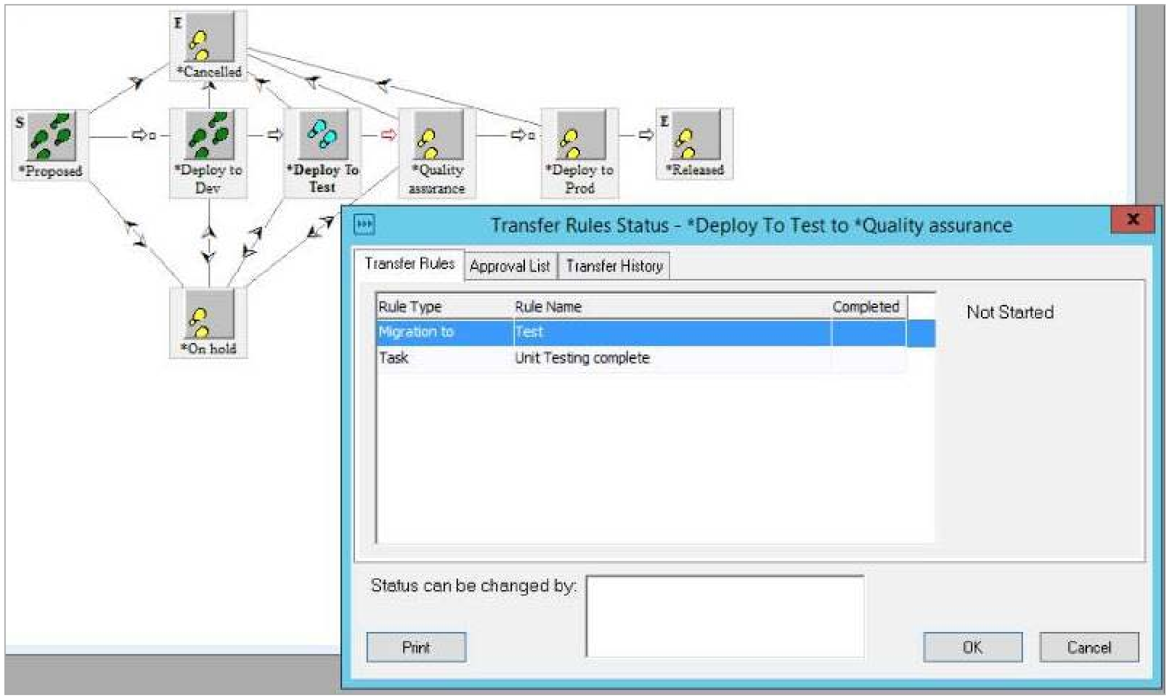


AND
Class: Manager. 2 optional user(s) must approve.
@ Fritz, Tim.
@ Gowdy, Mark.
Users: Individual Approver. 1 optional user(s) must approve.
@ Gowdy, Mark.

(Manager AND Individual Approver)

Status can be changed by:

Figure 2: Stat workflow with approval list



Rule Type	Rule Name	Completed
Migration to	Test	
Task	Unit Testing complete	

Not Started

Status can be changed by:

Figure 3: Stat workflow with transfer rules

Besides code and object changes, Stat **captures information about environment changes** that DevOps teams can use to seed regular reports. The teams can schedule and distribute those reports via email to stakeholders like compliance managers and auditors.

Stat and the Third Way

By helping to improve communication and reduce errors, Stat contributes to a **culture of high trust**. It reduces the need for privileged access to deploy code to test and production environments. When transparency and on-time delivery of releases become the norm, the relationship between developers and IT operations becomes smoother.

Even teams that have time for innovation are not always willing to take on additional risk. The key to creating a **culture of innovation and risk taking** is having a stable, well-defined and automated process for safely deploying applications and their changes, which the features in Stat support.

Conclusion

The goal of DevOps is to reduce the time between change request and change implementation, and to eliminate the unplanned work of break-fixing. That means removing any steps that do not contribute to that goal and automating others where possible, until developer and IT operations teams reach a defined and repeatable process. In the context of ERP, the trick is to accomplish that within the organization's tolerance for the risk that accompanies change.

Through the perspective of the Three Ways of DevOps, Stat helps ERP developers put in place the requisite control strength and audit reporting required for compliance with regulations like SOX.

Many ERP developers using the features built into Stat may already be doing DevOps and not know it. Many more ERP developers can easily use Stat to get on the inside track to DevOps adoption.

About the author

Dennis Nelson is a solution architect with Quest, as well as a seasoned IT professional and IT governance practitioner. He focuses on current trends and topics in ERP management, change management and IT governance, risk management and compliance (IT GRC). Dennis routinely presents as a conference speaker on topics such as organizational change management, application change management and IT GRC.

ABOUT QUEST

Quest provides software solutions for the rapidly-changing world of enterprise IT. We help simplify the challenges caused by data explosion, cloud expansion, hybrid datacenters, security threats and regulatory requirements. We're a global provider to 130,000 companies across 100 countries, including 95% of the Fortune 500 and 90% of the Global 1000. Since 1987, we've built a portfolio of solutions which now includes database management, data protection, identity and access management, Microsoft platform management and unified endpoint management. With Quest, organizations spend less time on IT administration and more time on business innovation. For more information, visit www.quest.com.

If you have any questions regarding your potential use of this material, contact:

Quest Software Inc.
Attn: LEGAL Dept
4 Polaris Way
Aliso Viejo, CA 92656

Refer to our website (www.quest.com) for regional and international office information.

Ebook-DevOpsCookbook-US-GM-39108

Quest