Mind the Gap

Six Steps to Bridge Software Development and Operations with Release Management

by Greg Hughes, CEO, Serena Software (now Micro Focus®)
# Table of Contents

<table>
<thead>
<tr>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>When Release Management Goes Wrong</td>
<td>1</td>
</tr>
<tr>
<td>Why Is Release Management So Hard?</td>
<td>2</td>
</tr>
<tr>
<td>Change Management: Necessary but Not Sufficient</td>
<td>3</td>
</tr>
<tr>
<td>DevOps: Beware the Purists</td>
<td>4</td>
</tr>
<tr>
<td>Continuous Delivery: Half of the Answer</td>
<td>4</td>
</tr>
<tr>
<td>Release Management Best Practices: Six Steps to Success</td>
<td>6</td>
</tr>
<tr>
<td>When Release Management Goes Right</td>
<td>9</td>
</tr>
<tr>
<td>Mind the Gap</td>
<td>10</td>
</tr>
<tr>
<td>Acknowledgements</td>
<td>11</td>
</tr>
<tr>
<td>Bibliography</td>
<td>11</td>
</tr>
</tbody>
</table>
All around us, software development is the engine of innovation. Software innovation disrupts existing business models by enabling products and services that are more intelligent and targeted. To ensure that they are the disruptors and not the disrupted, enterprises invest significant capital in managing the software development lifecycle (SDLC).

However, a gap in the SDLC is throttling back the velocity of new application innovation and causing the failure of critical business systems. This gap sits between development and operations—a traditional “no man’s land” of responsibility, tooling, and focus. Bridging this gap has to be a top priority for the modern enterprise.

For the highly regulated large enterprise (HRLE) in sectors such as financial services, insurance, healthcare, aerospace, defense and government, dealing with this gap is critical. A failed software release can destroy brands, reputations, regulatory standing, and other assets that can take years to rebuild.

While many enterprises have initiatives underway in applying DevOps, Information Technology Infrastructure Library (ITIL), or Continuous Delivery (CD) to bridge development and operations, these approaches fall short in the HRLE despite the zealous promises of their practitioners.

So, how should highly regulated large enterprises mind the gap between development and operations? The emerging discipline of release management provides the answer.

### When Release Management Goes Wrong

At 11:32 am on Wednesday, July 8, 2015, the New York Stock Exchange (NYSE) went down. Initially, many feared that the stoppage was due to a cyber attack on critical infrastructure in the U.S., and even President Obama was briefed on the issue*.

The source of the NYSE shutdown, however, was non-malicious and preventable—it was due to a failed release of new software. On the following day, as reported by the Wall Street Journal, the NYSE “confirmed an update to the exchange’s ‘matching engine’ led to communication problems between traders and the NYSE” (Hope, 2015). In the statement from the NYSE, the exchange stated that the “customer gateways were not loaded with the proper configuration compatible with the new release.”

---

* A few hours earlier, United Airlines grounded flights because of an unrelated failed router, adding to the anxiety of the situation.
When a problem with Release Management takes down one of the most important institutions for the smooth functioning of the world’s economy, then it is clearly a big issue for every enterprise.

Hardly a week goes by without a major media story of an enterprise failure due to a bad software release. Over the weekend of August 15–16, 2015, nearly 1,000 airline flights on the east coast of the U.S. were delayed or cancelled due to a failed software upgrade. Summer vacation travelers were stuck in airports for hours waiting for their flights. A defect in new functionality rolled out for the En Route Automation Modernization (ERAM) system caused the problem. ERAM had recently replaced a nearly 40-year-old system used by FAA Air Route Traffic Control centers (Brown, 2015).

Why Is Release Management So Hard?

Large IT organizations contain two very different tribes of people—the developer tribe and the operations tribe. These tribes are widely separated on many dimensions (physical location, skill set, methodologies/practices, experience, temperament, goals, and objectives).

When a software release goes wrong, the blame game starts.

The operations team says:

“We had no idea what was coming from development.”

Development says:

“It works on my machine!”

The responsibility for releasing new software stretches across two separate camps that see changes very differently and rely on separate systems for their work. Reconciling those differences is only possible through effective release management.
Change Management: Necessary but Not Sufficient

Many highly regulated large enterprises (HRLEs) have adopted the Information Technology Infrastructure Library (ITIL) process framework and implemented IT service management applications that include Change Management modules in order to support their operational processes. Though we often hear the terms “change management” and “release management” used interchangeably, they represent two distinct and important disciplines in the SDLC. Change management is about determining what changes are going to be accepted, implemented, and deployed. Release management is about deciding which changes should go together, tracking them through the lifecycle, providing visibility to those changes, and ultimately ensuring that they move safely into production.

Release management is concerned with the movement and implementation of the change request through the application development lifecycle. Generally, it begins when the release gets a name, e.g., 4th-Quarter Release. That’s when content—first the change requests and then the requirements, code, test scripts, test results, and sign-offs—gets assigned to the release and moves from development, to test, to systems integration testing (SIT), to user acceptance testing (UAT), to preproduction or staging, and then on to production.

The practice of release management ensures that the enterprise follows the release process through the entire systems development life cycle (SDLC)—milestones met, approvals obtained—and that the right artifacts move from stage to stage completely, safely, and speedily.

Without strong release management, HRLEs often find the management and communication of changes to be bewilderingly complex, especially given the huge volume of changes moving into production every week. By grouping a collection of smaller changes together in a set, release management vastly simplifies the complexity.
DevOps: Beware the Purists

Some organizations are experimenting with DevOps in order to improve the collaboration and coordination between development and operations, but DevOps is not the promised panacea for HRLEs. Pure DevOps preaches the blending of roles and sharing of responsibility between development and operations teams—in some cases creating “DevOps” units trained across their respective disciplines. However, in an HRLE there are important reasons to maintain the separation between development, QA, and operations. For example:

- **Regulatory compliance** requires the segregation of duties between development and operations. Segregation of duties is one of the most effective internal controls, and few HRLEs will tamper with the practice. Giving developers access to production is a nonstarter in an HRLE (Bird, 2014).

- **Benefits in scaling** can be obtained by consolidating the data centers and private clouds under a single, shared operations team with development aligned to specific business units.

- **Division of Labor** can result in better efficiency. Experts in operational or technical areas (e.g., lean, network security, Oracle DBA) can be more productive leveraging their focused skills rather than becoming “generalists” as prized by DevOps. At the scale of an HRLE, it is very hard to rely on such rare individuals. Let developers develop!

While some commentators have stated that DevOps “won’t work” for the large enterprise, it is not that simple (Shannon-Solomon, 2014). Many ideas coming out of the DevOps movement are valid and useful, but HRLEs should take particular care to separate what is applicable from what is damaging, costly, or impractical. The concept of a “DevOps Cookbook,” where HRLEs choose which specific ideas to apply in their environment, is intriguing (Kim, 2015).

Continuous Delivery: Half of the Answer

Continuous delivery is another popular practice for many organizations (Humble & Farley, 2011). A fundamental building block of continuous delivery is a “CD pipeline” that automates the progression of developed code across the SDLC into production—building, testing, staging, and promoting. Enterprises build a CD pipeline using tools to automate builds, deployment, and configuration management.
A working CD pipeline goes a long way to enabling improved release management. Automating steps in the SDLC eliminates manual labor, reduces errors, and improves repeatability. CD pipelines are very successful in companies built around a single application and a modern, simple infrastructure, e.g., Netflix, Etsy, and Box.

However, highly regulated large enterprises are vastly more complicated. They can have dozens to hundreds of software applications, crisscrossed with complex interdependencies, and running in a myriad of old and new environments (including mainframe systems). For example, Generali France, the second largest insurance firm in France, manages a total of 470 applications with a monthly average of 200 production deployments for the mainframe and 190 for distributed environments.

This complexity makes it critical for HRLEs adopting continuous delivery to complement their CD pipeline initiatives with a process management solution to control the approvals, governance, environments, and human workflows underlying release management. Process controls ensure that later stage environments (such as UAT, preproduction, and staging) are managed efficiently, correctly, and with accountability.

For both regulatory compliance reasons and internal visibility, HRLEs need to be able to trace a change in production back to the requirements, defects, or enhancements that triggered that change along with the corresponding source code updates. CD pipelines focus on the core artifacts (code, configuration), so a full solution needs to tie these artifacts together with the rest of the release content (e.g., requirements, enhancements, defects).

Because of the cost of failure, HRLEs need to maintain the “ripcord” of a rollback in case something goes very wrong in a release. CD purists often mandate the “fix forward” approach, which is not practical when downtime can bring the enterprise to its knees.
Release Management Best Practices: Six Steps to Success

If ITIL, DevOps, or continuous delivery initiatives are not enough, how should HRLEs mind the gap between development and operations?

The following six steps will help the HRLE master the best practices of release management and ensure the success of application deliveries. By comparing current practices to those described below, most organizations will find that there is more they can do to “mind the gap” between development and operations.

Step 1: Assign a Leader
The first step is to assign a clear leader. Any potential leader must be seen as a credible and honest broker who will listen to the issues and needs of all groups. What the enterprise needs is balance between the goals of development and operations, or to move fast without breaking things (Hughes, 2015).

If release reports to the development side of the house, the tendency is to be optimistic about the software changes and move them into production quickly, but quality and production stability can suffer as a result. If release reports to the operations side of the house, the tendency is to be more pessimistic and slow the delivery velocity down to allow more testing time, but production quality is higher. That is why release management in highly regulated large enterprises is often the responsibility of the quality assurance (QA) organization, which sits “between” development and operations.

Wherever the leader reports, he or she should have an understanding of the enterprise’s SDLC, the technical and process aspects of release management, and a good understanding of the enterprise’s tolerance for risk (i.e., striking the right balance between velocity and caution).

Most importantly, the leader needs support from senior executives and empowerment to drive change in development, QA, and operations practices.
Clear, measurable metrics help to unify the development and operations teams around a common goal.

**Step 2: Define Objectives and Metrics**

Second, define the objectives and metrics for the release management process. Start at the end goal and then work out how to get there. Define how to measure progress along the way.

Clear, measurable metrics help to unify the development and operations teams around a common goal. Executives will appreciate the newfound transparency into the software development lifecycle.

There are many possible ways to measure release success, and having a conversation across the enterprise about possible metrics will help develop a common vocabulary and commitment. For example, the following are all valid objectives:

- Increasing release (or change) throughput
- Reducing unscheduled releases
- Improving scope adherence
- Shortening lead time (the time from code commit to production)
- Increasing the quality of the end deliverable
- Improving awareness of release timing, schedule, and milestones

This step is complete when the leader achieves consensus around the reports and dashboards used for the release management process. Ultimately, the goal is to develop real-time data and reporting on the key metrics that measure the health of the release management process. The metrics serve two purposes: providing better visibility and understanding of the process, and enabling continuous improvement programs.

**Step 3: Map Out a Disciplined Process**

The next step is to map out a disciplined, collaborative release management process with defined workflows, gates, approvals, and information flow. Generally, there will be multiple tracks through the release process—for example, a fast one for a small emergency change and a slower one for major releases. Different business units across the enterprise frequently require their own specific adaptations as well.
Depending on the size of the release, many functional specialties in the HRLE (e.g., legal, compliance, security) and business executives will require sign-off. However, it is important to think carefully about who really needs to be in the approval process, as more approvers can needlessly slow the process, blur accountability, and reduce responsibility. Enterprises have found that eliminating unnecessary approval steps can improve both the quality and frequency of releases by focusing accountability on fewer people.

There are many possible conceptual frameworks for a release management process, including capability maturity model (CMM), CMM integration (CMMI), ITIL service transition, and continuous delivery (each with their own strengths and weaknesses). Typically, HRLEs must support multiple process types due to variation in maturity levels across the enterprise. To reduce wasteful activities, lean management practices are often applied.

Enterprises should plan to spend as much as six to eight weeks of effort on this step. The work of understanding existing processes and designing better ones is complicated and takes time.

**Step 4: Control the Content**

Step four is to get the content under control. That means developing a strategy to make sure that all application artifacts, not just source code, are in version control, and automation is used across as much of the application release process as possible. Inconsistency across environments—development, component testing, system integration testing, UAT, preproduction, and production—due to uncontrolled assets can cause needless thrashing. Worse, it can result in a failure in late stages of deployment.

HRLEs should build a single, hardened source code management system to make sure that their software assets are secure and controlled. Too many enterprises allow the proliferation of source code repositories, a situation made worse if based on open-source technologies such as GIT and Subversion that are designed by developers for developers, without the security and compliance required by the HRLE.
Step 5: Build the Supporting Infrastructure

Once the enterprise makes significant progress on the prior steps, the activity of choosing the tools to support release management can begin. Along with seeking specific functional criteria, prospective buyers should also ask the following questions of potential vendors:

- How much experience do they have supporting the special needs of highly regulated large enterprises? Does the vendor have a history of success in this complex domain?
- Do they provide “out-of-the-box” security and compliance with their products or is it something that needs to be added after the fact? Do the products scale across widely distributed enterprise teams vs. applying mainly to smaller workgroups?
- Do they provide an integrated set of products—across source code management, process control, application deployment automation—that will work together without extensive customization and maintenance? Do they work with the other tools in our SDLC? Does the vendor help enterprises across mainframe and open systems?

Step 6: Set Up Governance Approach

Finally, the organization needs to create a governance approach for release management. A well-functioning change management process including multiple, hierarchical change advisory boards (CABs) is a precondition of good release governance. The higher-tier CABs focus on exceptions, reviewing only the releases escalated by a lower change advisory board.

In order to function, the CABs need accurate information, a controlled process, and a living release calendar. So, getting everything else in order is required before the CABs can be effective. Most HRLEs have an established application used to support the change management process for operations. To enable real-time sharing of information between operations and release, the release management solution and the existing change management tool must interoperate.

When Release Management Goes Right

In enterprises that master the six steps, release management is a well-grooved process with the following attributes:

- Visible—the full status of upcoming software releases is clear to everyone.
- Controlled—each step in the process is enforced and tracked and exceptions are rare.
- Compliant—one system of record provides all data needed by auditors and regulators.
White Paper
Mind the Gap

- **Error-free**—software updates perform as required or are immediately recovered.
- **High-speed**—software gets released in hours and days rather than weeks and months.
- **Efficient**—nights and weekends are rare and automation is used extensively.
- **Secure**—the process, intellectual property, and application are protected from threats.

For example, Generali France had three goals for its release management initiative: “Reduce release complexity, shorten the deployment cycle so that it could be done in just a few clicks, and comply with new regulations,” according to Cyril Thenon, manager of the performance and industrialization skill center at Generali.

---

Software development needs to go faster in nearly all enterprises to compete in the modern world. However, the gap between development and operations often defeats this goal to speed the software development lifecycle. Bridging this gap is a top priority, especially for HRLEs.

In the highly regulated large enterprise, the gap between development and operations will not just go away—despite the optimistic pronouncements of the DevOps zealots, ITIL advocates, or continuous delivery believers. So, the better approach is to mind the gap by mastering the practice of release management. Following the six steps outlined in this paper will help enterprises start down this path with confidence.

Few enterprises have completed all six steps, which means that most organizations have room for continuous improvement. If your organization is just beginning to address release management, then the first step is to assign a leader. If you have a leader, then make sure that you are building consensus around the proper objectives and metrics. And so on with each of the six steps. Wherever you are on the journey to better release management, there is almost always more to do to mind the gap!
Acknowledgements

I could not have written this white paper without the help of my colleagues at Serena (now Micro Focus) who are experts in SDLC and release management processes. I am particularly grateful to Tom Clement, Julian Fish, Steve LeWarne, and Kevin Parker for their valuable editorial comments and input.

Bibliography


