Release Management: Bridging the IT Operations/Applications Development Divide

Serena Software
March, 2008
A Short History of Release Management ................................................................. 4

Why Do Our Release Management Practices Fail? .................................................. 4

The Intersection Between IT Operations and Applications Development .................. 5

Release Management Best Practices ........................................................................ 6

Serena Release Management Solutions ..................................................................... 8

Conclusion .................................................................................................................. 8
Executive Overview

How do you manage the traditional hand-off when Applications Development (App Dev) is ‘done’ writing code and needs IT Operations to put the changes into production? If you are like most organizations, this transition isn’t at all easy. Applications Development (App Dev), with their focus on customer satisfaction and aligning with business, wants it to be easy to change the production environment. A primary focus of IT Operations is on protecting the production environment and is often reluctant to introduce changes that could potentially introduce instability.

Both views are exaggerations and overly simplify the situation. But in those simplifications lies some truth. Organizations wanting to reduce the costs associated with putting changes into production, improve end-user satisfaction and business alignment, all the while reducing risks to the business caused by application failures, need to bridge this App Dev/IT Ops divide. These organizations need Release Management.
A Short History of Release Management

Forrester defines Release Management as, “The definition, support, and enforcement of processes for preparing software for deployment to production.”1 Release Management hasn’t always been a recognized discipline within either applications or IT Operations. Back when businesses were first investing in computers and software, Release Management was more akin to Release Free-for-all.

A developer would decide to change some code. He’d grab the printout of his COBOL program, mark up some changes, punch some cards to update the program and then push the code to production. The results were...not always optimal.

Forward a bit and you find some important changes. The coder would have to schedule some time to upgrade the production machine rather than just pushing in new modules whenever. In really forward-looking organizations, we might have a testing machine on which the developer could do some rudimentary testing before being allowed to change production. Still the results weren’t optimal and businesses, starting to depend on IT systems, demanded we do better.

Complex IT systems run modern enterprises today. Downtime or instability in these systems has a direct impact on revenue and profitability, and thus, changes to product systems must be closely controlled. Today, Applications Development organizations generally are not allowed access to the production system, or even the pre-production system, as a testing environment. In many cases IT Operations won’t even allow applications development access to these systems. App Dev also employ some sort of formal Applications Lifecycle Management (ALM) processes to help structure their development efforts. ALM’s goals are to improve quality, establish predictability in the software development process, improve the productivity of App Dev teams and establish effective change management.2

We’ve also made changes to IT Operations. Our IT Ops processes require that the end-users OK any production changes, and that R&D prove due diligence when it comes to quality. More and more often IT has adopted the IT Infrastructure Library (ITIL) framework to impose accountability and control over the production environment.

Why Do Our Release Management Practices Fail?

Although ALM has helped improve the quality of applications deployed to production, failures still occur. While IT Operations have made the production environment predictable and maintainable, it is often at the cost of agility. We have growing distrust between App Dev and IT Ops as their goals and expected performance measures become more and more at odds with each other. There is often no well-defined process for managing the hand-off between App Dev and IT Operations, causing confusion about what changes should be made to the production environment.

“They want Release Management, they just don’t know it yet,” says Benny Westaedt, He should know. Westaedt is both owner and Chief Architect for PSW, a consulting firm specializing in helping large banks fix their Release Management problems.3 He goes on to say that one of the biggest problems he has helping these organizations is that business and Applications Development have very different view of what a release is.

“Applications Development teams think a release is a specific version of an application that is ready for release,” said Westaedt. “The business doesn’t think in these terms, however. To the business, a release is a set of new capabilities that may improve operations, reduce costs or help drive new business. In actuality, a release may span many new systems, include both hardware and software, and require infrastructure upgrades.” He goes on to say that, “This difference in perception is at the root of many problems faced by businesses trying to fix their Release Management problems.”

Besides the issues noted by Westaedt, there are others factors that contribute to the failure of our current Release Management processes.

---

2 “What is the application lifecycle” Ovum, September 2007.
3 http://www.it2change.be/.
• **COMPLEX BUSINESS ENVIRONMENTS**: Business today is complicated. Operations span the globe, requiring adherence to myriad regulations, sensitivity to multiple cultures, and awareness of political nuances. In the United States alone, public banks are subject to Sarbanes-Oxley, Basel II, OFAC, BSA and Patriot Act regulations. In this environment, unauthorized or high-risk changes to production environments can cause not just loss of revenue, but jail time for corporate officers.

• **COMPLEX ARCHITECTURES**: It started when client/server architectures and three-tier architectures became all the rage. Suddenly an application wasn’t just a bunch of modules. A release could include dependencies on operating systems, databases, 3rd party applications and client environments. Today we have SOA-based applications that may be spread across many clients and servers on both sides of the corporate firewall. It isn’t always easy to know what parts and pieces of the production environment will be affected by a change to a production system.

• **HIGHLY DISTRIBUTED ENVIRONMENTS**: Long gone are the days when a few coders, who all knew each other, could walk down the hall to talk to a colleague in IT Ops to get their changes deployed into production. Today’s App Dev and IT Operations are highly distributed, work 24/7, and may be internal employees, contractors or offshore resources. An App Dev team could have some members in NY and San Francisco, an offshore development center in India and a QA team in Singapore. IT could be spread around the globe, with a data center in Omaha, a support center in Ireland and contractors providing many IT services. Not only are applications more complicated, but the production environment is now more complicated as well. It’s not always obvious who has responsibility for tasks, and where the various parts and pieces of infrastructure reside.

• **DIVERGENT PRIORITIES**: There has always been a tug between these two groups, but today the differences are more pronounced. App Dev is trying to become more agile and customer-centric. They are under pressure from the business to deliver more capabilities faster, with the ever looming threat of outsourcing and off-shoring.

IT Operations on the other hand, has been under different pressures. They have been concentrating on reducing costs and consolidating data centers. Their bonuses have been tied to SLAs, system availability and production system stability. The result is that App Dev wants to deliver as much as possible as quickly as possible, and IT would be happy if nothing ever changed.

In Forrester’s words, “…Naturally, business customers will always ask for as much functionality as possible to be delivered as soon as possible. This becomes especially problematic when their app dev partners are so focused on customer satisfaction that they are unable or unwilling to say no and push back on the requested scope and timeline.”

While all the complexities listed above contribute to ineffective Release Management, it is the divide between App Dev and IT Operations that is the most problematic. Given a cohesive team with unified goals properly motivated, they will be able to overcome the problems created by complex business and development environment. However, when IT Operations and App Dev are divided, Release Management will likely fail even in the simplest environment.

If we are going to put effective Release Management in place, we need to establish common ground between IT Ops and App Dev. We’ve got to find a way to have both work together for a common goal and eliminate even the idea that we ‘hand’ off from App Dev to IT. Instead, both must work together to bring a release into production.

### The Intersection Between IT Operations and Applications Development

Applications Development and IT Operations start from very different perspectives. Not only do they have different management frameworks, they have competing motivations, as we’ve already discussed. It is possible to find common ground. While not all applications development organizations employ CMMI for their process improvement framework, enough do that we can use it here as an example. Similarly, while not all IT organizations use ITIL as their Service Management framework, enough do that we can use ITIL as a

---

representative IT Operations framework. If we can find common ground between ITIL and CMMI, we will have made a good start bringing IT Operations and App Dev together.

- **CMMI PRODUCT INTEGRATION**: In CMMI for Development, “The purpose of Product Integration (PI) is to assemble the product from the product components, ensure that the product, as integrated, functions properly, and deliver the product.” In other words, Product Integration is where Release Management happens. You can review the goals and practices of CMMI Product Integration for details, but in a nutshell they are: plan for the release, test and verify the release, and deliver the release.

- **ITIL RELEASE MANAGEMENT**: In ITIL, Release Management is where Release Management happens. You can review the specific activities associated with ITIL Release Management, but in a nutshell they are: plan for the release, test and verify the release, and deliver the release.

Given the goals and practices of both CMMI Product Integration and ITIL Release Management are the same, it should be easy to reconcile the two. Such is not the case, however. As Gartner notes, “...Neither ITIL nor CMMI governing bodies have provided or are developing definitive guidance about how to harmonize process definitions, or how to streamline the hand-offs from development to operations for change management process workflows.”

Never fear. While the governing bodies haven’t stepped up, practitioners have. Let’s take a look at some best practices for Release Management that leverage both ALM disciplines within the CMMI framework and IT Operations disciplines within the ITIL framework.

**Release Management Best Practices**

You need to be realistic about Release Management. Release Management won’t happen overnight. Garter estimates a consolidated IT Change Management process, admittedly a superset of Release Management, “…will require three to five years of mutual investments by development and operations in process mapping, gap analysis and reconciliation.”

As you work towards implementing effective Release Management, don’t be frustrated. Get started in the right direction, be willing to make mistakes and learn from your mistakes, and make your policies flexible enough to change with changing business and technology environments.

**PLAN THE RELEASE**

**Set up a Release Management Team.** Put together a cross-functional team appropriate for the release in question. Team members may shift depending on the type of release (Major, patch, exception) the product being released and the risk associated with the release. The Team should be cross-functional, having members from the business, IT Operations and Applications Development.

Build a strong team and empower it to make decisions. The team needs to have not only executive support on paper, but executive backing when it makes hard decisions. As Forrester says, “...The more often business customers and executives overrule the release management team, the less useful the release management team becomes.”

The team must also have a chair, someone responsible for making sure that things get done, roughly equivalent to a Project Manager. Given the tension between App Dev and IT Ops, the chairperson should come from the PMO or some other neutral organization.

**Publish a Release Plan.** The release team must know exactly what is to change in the production environment and who is to make the changes. The team should document all hardware, software or other infrastructure

---

5. “How to Navigate IT Change From Development to Production” Gartner Research, October 2007.
dependencies. Finally, the team should have contingency plan in place so the release can be rolled back in case of failure. The plan should be made public so everyone knows who is responsible for taking actions, and when those actions are to take place.

- User Training
- Internal and External Marketing
- Documentation
- Hardware Dependencies
- Software Dependencies
- Configuration Changes
- New or Modified Client Software
- New or Modified Server Software
- New or Modified Middleware
- Database Changes
- App Server Changes
- Integrations
- User Acceptance Testing

**Conduct a Retrospective.** Take a hint from Agile software development. A retrospective is a useful tool for understanding what was good or bad about the previous release. Keep the retrospective positive. Squash any blame-game antics and concentrate on making sure the next iteration is better than the last. Adjust your process as necessary to incorporate what you learned from the last release.

**TEST AND VERIFY**

**Set Quality Expectations Appropriate to the Release.** Is it realistic to expect zero defects and exhaustive testing? If the release is to a mission critical application, exhaustive testing is expected, since defects can be highly costly to the business. If the release is an update to an internal application, enterprises may have a higher tolerance for defects and do less testing. The release team must have the authority to define quality standards that must be met before a release goes into production. Compromising on quality can have huge negative impact on a business, but unrealistic expectations are just as damaging, leading to a general disregard for quality standards. Remember the national 55 MPH speed limit? Back then nobody paid attention to the law.

Most organizations have a mix of internally built custom applications with old monolithic architectures, packaged applications with custom configurations, SaaS applications, new SOA-based applications, portals, BPM solutions and even mashups and composite applications. It is no longer easy to understand risk associated with application changes. Good Release Management comes from good information, and that means understanding your architecture and the use to which each application is put.

**Enforce End-User Acceptance Testing.** Effective Release Management includes the concept of separation of duties. Only End-users have the authority to decide whether an application is ready for production, not App Dev or QA. If the business balks at providing the resources for end-user acceptance testing, then publish the risks and let everyone know the release is at risk. There is no substitute for end-user acceptance testing.

**Automate Wherever Possible.** Testing can be tedious, especially regression testing of a large number of features. Automation provides welcome relief from tedious testing tasks, and reduces the risk that severe defects will creep into production due to inattention or human error.

**DELIVER THE RELEASE**

**Communicate to Stakeholders.** Let everyone know when you will be moving the release into production. This may actually be more difficult than it sounds. If you are making changes to SOA-based services, you may not know who your consumers are. If you are upgrading a server OS, you may now know everyone who is using the server. Does the release require clients to upgrade their browsers? Some users won’t get the message, or won’t understand that the change applies to them. Don’t be discouraged. Do the best you can, and when you get complaints, which you will, save them for the retrospective so you can do better next time.

Make release status transparent. Knowing the bad news is better than being kept in the dark. We all want to publish success and hide failure. It’s human nature. It won’t help you in this case. Everyone will still find out, and you will lose credibility. Go ahead and make your successes and failures known.
Automate wherever possible. Humans are human. We make mistakes. If you take the grunt-work out of release management and automate where possible, you lessen mistakes, and keep people free to make exception decisions. Be sure to automate both the deployment to production and the rollback from production.

Ensure Stakeholder Readiness. Release Management isn’t just about putting the software into production. If users need to be trained, make sure the training happens before you deploy. Coordinate with stakeholders to make sure the timing is right. Imagine what would happen to your CIO if you decide to change the financial system in the last quarter, or update the CRM system the last week of the quarter. If you have the right team these issues should be identified in the planning stage, but be sure to double-check before you go live.

One final note about Release Management. The amount of governance and oversight you put in place should depend on the type of release, how much risk your organization is willing to accept into the production environment, history of past failures and how well your teams work together. Too much governance can be just as damaging as not enough, so start with a light hand and evolve over time, putting in just enough oversight and management to control risks and ensure success.

Serena Release Management Solutions
Release Management isn’t something you buy, it’s something you do. However, the right tools can help you put effective Release Management in place. Serena Software has a complete line of products that will help you automate your Release Management processes.

Serena Dimensions CM lets Release Managers automate building and deploying applications for release across distributed systems. Dimensions CM also includes Change Management so stakeholders know what changes are in a release.

Serena ChangeMan ZMF is the world’s best Change and Configuration Management solution for mainframe development and deployment. ChangeMan ZMF allows Release Managers control over changes, approvals and deployment of mainframe modules into the production environment.

Serena Mariner gives the Release Management team the tools it needs to practice effective project and portfolio management. With Mariner, teams can not only publish a release project plan, but can also provide visibility into the entire project portfolio so everyone understands priorities, budgets and schedules of the various projects in the release pipeline.

Serena Business Mashups 2008 ties all your Release Management processes together. Serena Business Mashups will let you automate your release approval process, your stakeholder notifications, and even your release deployment. Release Managers won’t need to jump from tool to tool to get the job done. Instead, you can mash up your various tools into a unified and automated Release Management Process.

Conclusion
Businesses have been doing Release Management ever since the first software was written for the first computer used in a commercial operation. Release Management as a formal discipline is new, fueled by ever-more complex business, technical and organizational environments. Additionally, the divide between the motivations and goals of Applications Development and IT Operations is making the traditional hand-off between these two organizations less effective.

Establishing formal Release Management with members from IT Operations, Applications Development and Business will help bring these organizations back together. Although frameworks such as CMMI and ITIL haven’t tackled the difficulties of establishing Release Management across the IT Ops/App Dev divide, practitioners have. With a few common-sense guidelines and a willingness to improve over time, your organizations can also put effective Release Management in place.
ABOUT SERENA
Serena Software, Inc. is a global leader in Business Mashups and Application Lifecycle Management (ALM), for distributed and mainframe systems. More than 15,000 organizations around the world, including 96 of the Fortune 100, rely on Serena solutions to automate the application development process and effectively manage their IT portfolio. Serena is headquartered in San Mateo, California, and has offices throughout the United States, Europe, and Asia Pacific. For more information on Serena solutions and services, visit www.serena.com.

CONTACT
Learn more about the enterprise-wide power of Serena products by visiting www.serena.com or contacting one of our sales representatives in your area.

Serena Worldwide Headquarters
Serena Software, Inc.
Corporate Offices
2755 Campus Drive
Third Floor
San Mateo, California 94403-2538
United States
800.457.3736 T
650.522.6699 F
info@serena.com

Serena European Headquarters
Serena Software Europe Ltd.
Hertfordshire
Abbey View Everard Close
St. Albans
Hertfordshire AL 1 2PS
United Kingdom
+44 (0)800.328.0243 T
+44 (0)1727.869.804 F
ukinfo@serena.com

Serena Asia Pacific Headquarters
Serena Software Pte Ltd.
360 Orchard Road
#12-10
International Building
Singapore 238869
+65 6834.9880 T
+65 6836.3119 F
apinfo@serena.com

Copyright © 2008 Serena Software, Inc. All rights reserved. Serena is a registered trademark and Mashup Composer and Mashup Exchange are trademarks of Serena Software. All other product or company names are used for identification purposes only, and may be trademarks of their respective owners. Sept07