

## **Empowering DevOps: Implementing a Real Application Release Automation Solution**

## Abstract

Application release is a key IT process—one that consumes large amounts of time and resources and is critical to a business' success. Yet, in contrast to the many functions that IT organizations have automated, application release still typically involves a great deal of manual effort. As a result, it's both time-consuming and error-prone—plus the vast majority of releases miss their due dates.

A good application release automation (ARA) solution can solve these problems—but not all ARA tools are capable of meeting all the challenges inherent in automating this highly complex process. This white paper identifies the key challenges, discusses their business impact, and explains the capabilities that an ARA solution must have in order to meet the needs of today's enterprises.

## Contents

Overview.....	3
What Are the Challenges?.....	3
What About Scripts?.....	5
Business Impact.....	5
What About Compliance and Control?.....	6
What Makes a Good ARA Solution?.....	7
End-to-End Automation.....	7
Planning.....	7
Control.....	8
Deployment.....	8
Support for Real-Life Enterprise Environments.....	9
Scalability.....	9
Reliability.....	9
Flexibility.....	9
Security and Compliance.....	9
Security and Control.....	10
Auditing and Reporting.....	10
Complex Event Processing.....	11
UC4 Application Release Automation: The Most Complete Solution.....	11
End-to-end Process Automation.....	11
Support for Real-Life Enterprise Environments.....	12
Security and Compliance.....	12
Complex Event Processing.....	13
Business Impact.....	13
Conclusion.....	14

## Overview

Automation has helped IT departments simplify many complex processes—from job scheduling to virtualization to file transfers. In so doing, it has increased their efficiency, reduced their operating costs, and eliminated the errors often associated with manual processes.

But one key area typically still involves a great deal of manual effort: application release. For a single application, the process may involve six or seven people—a developer or two, a QA manager, an application release manager, a network administrator, a middleware administrator, and a database administrator. The setup required to deploy an application—whether to QA or production—can take anywhere from 30 minutes to several hours, depending on the size and complexity of the application. And this tedious, time-consuming, and error-prone process may have to be repeated 50—or even 100—times over the course of a few days as the application is readied for production.

As a leading analyst notes, “Release management must be tightly coordinated across application and production control teams, reconciling the tensions and conflicts between the goal of introducing new or changed functions, and the challenge of minimizing risk and maximizing quality of service.”

The extensive manual effort typically involved in this task is why we refer to the application release process as “the next frontier” of IT automation. By fully automating this next frontier, end to end, organizations can reduce errors, lower their costs, create a process that’s more easily tracked and audited, and get their applications to market much faster.

## What Are the Challenges?

Current approaches to application release—whether fully manual or semi-manual (i.e., script-based) have a number of inherent challenges:

- **Environment diversity.** It’s typical to have separate teams responsible for deploying the application into each operating environment—and each team may work entirely independently of the others, leading to inconsistencies between them. The dynamic nature of cloud and virtualized environments makes matters even more complicated. And the fact that each team—development, QA, and operations—works on its own, with little communication between them, makes it hard to deal with any issues that arise.
- **Lack of integration.** Each IT area—development, QA, and production—typically operates as a silo, using its own tools, with very little integration between tools used by different areas. For example, developers may use Subversion; QA may use HP Quality Center; and production staff

may use Maven. The lack of integration between these tools leads to further inefficiencies, errors, and long communication cycles. In addition, instead of the release process flowing smoothly from one environment to the next, people must move it along manually—for example, sending an email to initiate the next step rather than having it automatically follow the previous step.

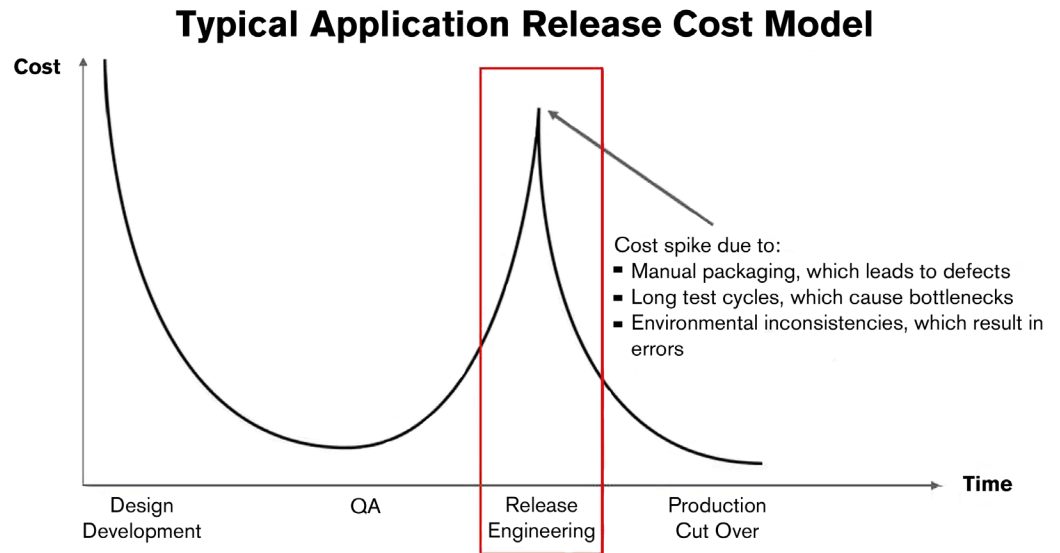
This lack of integration is particularly problematic in view of the frequent interdependencies between applications. For example, if Application A consumes a service from Application B, and you update Application B, will Application A still work?

- **Knowledge management difficulties.** With so many people across the organization working on application updates, and each group typically keeping its information in its own set of spreadsheets, organizations have no good way to manage application knowledge across the enterprise. There's no traceability when something goes wrong—no way to find out who was responsible for that part of the application.
- **Update frequency.** Developers are under pressure from the business side of the organization to get updates into production quickly, so end users can have access to the new features. Operations groups, on the other hand, focus on factors like control, security, and manageability, which tend to slow things down. Still, the trend is towards more frequent updates. In some cases, production updates may occur as often as daily—particularly in the case of dynamic Web applications. In other cases, they may be weekly or monthly, or, more rarely, quarterly.

Regardless, in a large organization running many hundreds of applications, with each application running on dozens, or even hundreds, of servers and updated, on average, once a week, the amount of manual effort required by current approaches ends up taking huge amounts of staff time, costing a lot of money, and significantly extending the timeframe required to get new applications to market.

- **Conflicts between development, QA, and operations teams.** The operations side typically has a fixed window established for deployment into production—yet developers may take longer than planned to get the updated application into testing. As a result, the QA team is caught in the middle, under pressure to both deliver high quality and to complete the testing in less time than they really need.

Figure 1 illustrates the impact of these challenges on the costs involved in application release.



*Figure 1. Manual application release processes typically lead to high release engineering costs to correct defects and errors—problems that wouldn't arise with a release process that's automated end to end.*

## What About Scripts?

Many IT departments have taken steps to automate part of the application release process by writing scripts to handle some of the tasks involved—for instance, to copy the necessary files and perform some of the required configurations. But not only are such scripts limited as to how much of the process they can automate, but they can also introduce their own problems. For instance, they are extremely time-consuming to write and maintain; they are often error-prone; and, because different individuals may have been involved in writing them, they may contain problematic inconsistencies.

The lack of built-in controls in scripts also makes them a management nightmare. It is difficult, if not impossible, to trace script execution in a production environment. And because scripts may access highly sensitive application data, this difficulty presents not only an IT management issue, but also a significant security threat.

## Business Impact

The application release issues described above have significant impact on the business's bottom line. Our research—and our customers' experiences—indicate that both manual and semi-manual application release processes take an inordinate amount of time from multiple IT groups:

- Development teams typically spend 30% to 60% of their time in the

application release process.

- QA organizations spend 50% of their time building, configuring, and maintaining test environments.
- Operations teams spend 75% of their time on application release.
- Support teams spend 75% of their time on application release problems.

These approaches also result in an inordinate number of problems:

- 50% of all application issues are a result of differences between test and production environments, creating false negatives during QA testing—and 90% of these environmental problems could be prevented by application release automation.
- 60% of data center application failures are the result of faulty deployment and erroneous configuration changes.
- Application problems typically increase by 200% in the first 10 days following a release—and, because of lack of communication and controls, it's often hard to tell if the problem is due to an environmental factor or an actual application bug.

All of these factors not only add significantly to IT operating costs, but also delay time to market. In fact, among companies that use either manual or script-based deployment, release delays are so much a fact of life that:

- 78% of application releases miss their original due date.

In the case of internal releases, these delays result in lost productivity, as business users must wait longer to access the new features. In the case of external releases, these delays result in loss of revenues. Either way, the effects are felt in the company's bottom line.

And then there's the human factor to consider. The issues described in the sections above are very frustrating for all involved. The errors and delays they generate lead to a great deal of blame and finger-pointing—mostly to no avail, because the same problems will happen again with the next release. Ask any IT person—regardless of whether they're in development, QA, or production—what aspect of the application life cycle they dislike the most, and the answer will generally be, "Deployment."

And here's the kicker:

- 90% of all these problems could be prevented by application release automation.

### **What About Compliance and Control?**

There's one additional problem with manual and semi-manual methods, and that

is the difficulty of ensuring regulatory compliance and tracking changes. From a legal standpoint (e.g., ensuring compliance with such regulations as HIPAA, Sarbanes-Oxley, Gramm-Leach-Bliley, PCI, and Basel II/III) as well as from an auditing perspective, it's important to be able to control who has access to what—and to know the source of any changes that are made.

For example, from both a stability and a fraud prevention viewpoint, it's important that developers not be allowed direct access to production environments. But if anyone can run a script, there's no way to keep changes from being made to production systems. And whether the application release process is completely manual or semi-manual (i.e., script-based), there's no way to track what changes were made—or who made them.

## What Makes a Good ARA Solution?

Releasing an application into production involves a lot more than taking a package from Server A and moving it to Server B. An effective application release automation (ARA) solution must provide end-to-end automation, beginning with the critical planning stage. It also must be capable of supporting real-life enterprise environments, and therefore requires extensive scalability, reliability, and flexibility. It must provide a high degree of security and compliance. And if it also includes complex event processing, that's a real plus, as that means the solution will be able to recognize problems and respond to them proactively—before end users are even aware of them—by rolling back to the previous release.

Let's look at each of these requirements in more detail.

### End-to-End Automation

A true end-to-end solution encompasses planning and control steps, as well as deployment automation. Automating all three aspects of the application release process is critical to ensuring a smooth, efficient deployment.

### Planning

Many teams are involved in the application release process—both across different environments (development, QA, operations) and within each environment. The benefit of a formal planning process is that it provides the necessary visibility ahead of time, so everyone involved know what's coming and can plan for it. Planning also helps identify resource conflicts between parallel projects and provides the basis for project tracking and reporting.

The planning step includes planning both the release itself (that is, the applications and components it is to include) and the environment into which it is to be released. Key points to determine at this stage include:

- Establishing server requirements for QA testing and production
- Scheduling testing time in advance to make sure the necessary resources will be available
- Determining the release process—who owns which tasks and who needs to authorize which steps

### **Control**

The entire application release process needs to have extensive controls in place in order to provide consistency across the entire application life cycle, from development to production. These controls will ensure that:

- Only those who are authorized to perform a task, in both QA and production environments, can actually do so.
- Nothing is deployed into production until the appropriate team has signed off on it.
- The application environment is centrally managed and is consistent.

This last point plays a huge role in minimizing delays. If someone tweaks the environment during the release process and that change is not documented in a central location available to all involved, the application won't work when it gets to production—and troubleshooting it to find out what broke and why can take a great deal of time.

The solution is to build templates specifying all the environmental details. These templates provide centralized control, ensuring that the application will continue to work as it moves from one environment to the next.

### **Deployment**

To fully automate application deployment activities while avoiding all the pitfalls, you need a highly sophisticated automation engine that can:

- Deploy applications across multiple platforms
- Deal with dependencies between applications
- Interact with multiple versions of the middleware
- Check to make sure the production environment matches the requirements before pushing the software out
- Roll back the deployment if something breaks during the rollout

For ease of reuse, the solution should also be object-oriented. That way, the same release process can easily be used across development, QA, and

production environments, with the only change being the information that's being passed. So, for example, if you set up a workflow for releasing Oracle database applications, you can use it for all Oracle database applications, across all environments. In contrast, with a script-based application release environment, the script has to be rewritten for each environment—an approach that not only is time-consuming, but also error-prone.

### **Support for Real-Life Enterprise Environments**

Some ARA solutions work fine in small organizations, but aren't up to handling the complexity of enterprise environments, which may have thousands of applications and tens of thousands of servers to manage. To work for large enterprises, an ARA solution needs extensive scalability, reliability, and flexibility.

#### **Scalability**

Some ARA solutions require multiple separate deployment engines to release applications to the thousands of servers typically found in large enterprises, such as financial and healthcare organizations. However, such an inefficient approach cancels out many of the benefits inherent in automating the release process, and can turn the process into a maintenance nightmare. A truly enterprise-capable ARA solution requires only one deployment engine, no matter how many servers are involved.

#### **Reliability**

When you're dealing with a mission-critical application, you want the release infrastructure to be as solid as the one used for running the application. This means you need a robust deployment engine that supports clustering and disaster recovery scenarios (such as failover to different locations).

#### **Flexibility**

To keep everyone in your IT organization happy, you need an ARA solution that can be adapted to suit your organization's processes, terminology, and tools. You want to be able to adjust it as needed to meet the needs of any teams that operate it—whether they are developers working on preproduction versions of the application, QA staff testing it, or operations staff releasing it into production. In short, you want to be able to customize the tool to fit your processes, rather than modifying your processes to fit the tool.

#### **Security and Compliance**

To meet security requirements and ensure compliance with applicable regulations, an ARA must offer fine-grained, centralized controls and also

provide centralized reporting and audit trails.

### **Security and Control**

Good security requires having a granular view of who has access to what—and a way of making sure that team members have access only to what they need to do their jobs. So, for instance, if an individual is responsible for only 10 out of 1,500 applications, there's no need for that person to have access to the other 1,490.

Centralization of control is also critical. Instead of having a dozen people, each with his/her own set of user accounts and credentials, it's much more efficient (and secure) to have a single release platform for managing the entire release process, with a single set of user accounts and credentials. That way, instead of a release requiring the involvement of a developer or two, a QA manager, an application release manager, a network administrator, a middleware administrator, a database administrator, and possibly a security administrator, you've got a standardized release process, stored as a template in the ARA solution, that can be kicked off by a single application release manager, with a single set of credentials.

It's also important that the controls be able to limit what aspect of the application release cycle individuals are allowed to access—as opposed to allowing anyone who logs into the application release tool access to any part of the process. For example, an organization may want to be able to match read/write access to the business process an individual is performing. In addition, it may want to apply controls on a per-application basis, a per-environment basis, or even a per-server basis.

A good ARA tool will provide a centralized set of controls that let organizations establish credentials with whatever degree of granularity they desire.

### **Auditing and Reporting**

Another reason for having a centralized platform for managing the application release process is that it makes it easy to track changes. With all information in one place, if the application breaks, you have an audit trail that you can use to locate the exact point at which the problem was introduced—and also determine who made that change.

A centralized platform also serves as the basis for centralized reporting, with all the analytics you need to find out anything you want to know about your application release process—from how many applications were released in any given period and what resources were used, to how long each release took and what percentage of your applications were released on time.

### Complex Event Processing

An ARA solution that includes a complex event processing (CEP) engine offers additional benefits beyond the basic requirements listed above. The CEP capabilities provide a layer of intelligence that can track disk I/O, memory utilization, CPU utilization, and thousands of other performance-related factors; and, if it spots a problem, automatically roll back the application to the prior release—all before your end users or customers are aware of any performance degradation.

### UC4 Application Release Automation: The Most Complete Solution

UC4 Application Release Automation automates the entire release process, end to end—from planning and control to automated deployment and monitoring. It's the most scalable solution available today, able to handle even the largest deployments, and supporting more databases, operating systems and tools than any other solution. It gives you complete visibility and control across all application releases, enterprise-wide, with extensive security, fine-grained control, and robust auditing and reporting capabilities. And it includes a powerful CEP engine that can foresee performance problems before users are aware of them and automatically roll back the release to the previous build.

### End-to-end Process Automation

UC4 Application Release Automation automates the entire release process, from development to production—and beyond, throughout the application's life cycle. It automates:

- **Planning.** With UC4's solution, you can plan all aspects of the release: the environment, servers, components, dependencies, configurations, and release milestones. And with complete visibility into the release process, it's easy to manage and track the application release workflow.
- **Control.** UC4's solution eliminates the majority of errors that crop up in testing and production by ensuring a consistent environment throughout the application delivery process. It lets you create templates that make it easy to centrally manage environmental changes and configurations, ensuring a consistent environment from development through to production. This capability alone significantly reduces errors.
- **Deployment.** UC4 Application Release Automation fully automates deployment across QA and production environments.
- **Operations.** The automation provided by the UC4 automation engine

doesn't stop when the application enters production. It continues through the application's entire life cycle, automating such tasks as job scheduling, application processes, managed file transfer, virtualization, and run books. And it does all this with the same underlying automation engine—which is why we call our approach ONE Automation. With UC4, you get a single, unified automation solution capable of intelligently orchestrating all your business processes, applications, and IT infrastructure.

### Support for Real-Life Enterprise Environments

UC4 Application Release Automation offers the scalability, reliability, and flexibility that large enterprises need:

- **Scalability.** A single installation of the UC4 automation engine can handle millions of tasks. Being able to do all this with just one engine means lower costs for you, plus simplified maintenance.
- **Reliability.** The UC4 automation engine is battle-proven: it's been used by more than 2,000 customers for over 20 years.
- **Flexibility.** Every company has its own ways of doing things. With UC4 Application Release Automation, you can customize everything from terminology to the technical requirements, to match the way your company works.

### Security and Compliance

UC4 Application Release Automation enables you to fully address all security and compliance needs:

- **Security and control.** Controlling who has access to what aspects of an application is critical both to security and to all types of regulatory compliance—from HIPAA to Sarbanes-Oxley. UC4 Application Release Automation gives you extremely granular control, enabling you to restrict who can view which parts of an application, who can make changes, and who can deploy the application into which environments (development, QA, or production).
- **Auditing.** UC4 Application Release Automation provides you with a complete audit trail of every activity that takes place—not only during the application release process, but also afterwards, when the application is in production. And because event logging is centralized, you can respond to an audit request easily, without having to pull log files from a dozen different locations.
- **Reporting.** With UC4 Application Release Automation, you get

customizable reporting that can be tailored to the needs of each audience—developers, QA staff, operations, and business leadership.

### Complex Event Processing

UC4's solution delivers the additional bonus of a powerful CEP engine—one with real-time monitoring capabilities that enable it to identify problems in a new release and roll it back to the previous build before users are even aware of any performance degradation.

### Business Impact

By delivering true end-to-end automation, the ability to support real-life enterprise environments, granular security and compliance features, and CEP capabilities, UC4 Application Release Automation will benefit your organization in many ways:

- **Speed.** The ability to release applications much faster than before means you'll be able to get customer-facing applications to market sooner, increasing your competitive advantage. You'll be able to deliver all the new functionality your business users want—in every release cycle. You'll have time to do more testing than you can now. You can build your releases continuously and release them in a single step to physical, virtual, and cloud environments. Market acceptance testing of new applications will also be faster. And you'll be able to make 100% of your release dates on time, instead of missing 78% of them.
- **Planning, reporting, and control.** With centralized planning, your teams will be better prepared to process new releases and can make sure the necessary infrastructure capacity is available when it's needed. Centralized control will reduce audit risk, simplify troubleshooting, and eliminate finger-pointing between teams, because it will always be clear who made what changes. And centralized reporting will let you verify compliance with all relevant regulations.
- **Lower costs, fewer errors.** An end-to-end application release automation solution will reduce manual efforts by 80% or more; reduce audit and remediation cycle times by as much as 55%; and virtually eliminate application release errors. All of these factors will significantly lower your application release costs.
- **Enhanced productivity.** By automating release functions that are now performed manually, an ARA solution can more than double your team's productivity—without adding any headcount.

In addition, because the same automation engine that powers UC4 Application

Release Automation also includes powerful CEP capabilities, you'll be able to release new applications without worrying about performance degradation. The solution's intelligence enables it to monitor thousands of performance-related factors in real time and roll the application back to the previous build whenever necessary—and do so early enough that your end users and customers won't even be aware of a problem.

## Conclusion

Application release is a core business process for every organization. It spans all IT groups—development, testing, and operations. Yet, in many IT organizations, this important process is mostly manual—an approach that's both costly and time consuming.

By moving up to a true end-to-end ARA solution—one that can support even the largest enterprises with all the functionality they need—you can save time and money, significantly reduce errors, enhance your team's productivity, and get all your releases out on time—every time.

For more information on what UC4 Application Release Automation can do for you, go to <http://www.uc4.com/what-we-do/it-automation/application-release-automation.html>, e-mail us at [info@uc4.com](mailto:info@uc4.com), or call us:

In the US at (877) 464-7300 (toll-free)

In Europe at +43 2233 77880