Use DevOps And Supply Chain Principles To Automate Application Delivery Governance

Processes: The Modern Application Delivery Playbook

by Diego Lo Giudice

November 10, 2016

Why Read This Report

Today's application development and delivery (AD&D) leaders need speed and quality at a low cost to create software that competes for customers. Agile and DevOps practices enable companies to achieve all three, but paralysis over compliance, audit, or security concerns causes many to cling to manual governance practices that are ineffective even at slower speeds. This report describes how application delivery organizations are applying automated supply chain management practices to improve both application delivery governance and business results.

This is an update of a previously published report; Forrester reviews and updates it periodically to ensure continued relevance.

Key Takeaways

Don't Govern Broken Processes; Fix Them First

Leading software organizations make it easier for teams to do the right thing instead of merely defining standards and then measuring to see if teams follow them. They deliver faster, their quality is higher, and everyone is more effective.

DevOps Practices Improve Transparency And Visibility

Automation provides the objective information — and visibility into that information — that organizations need to make application delivery decisions. In place of periodic, subjective, and time-consuming status reports, firms get real-time insight into application health and delivery progress, gathered as a natural part of the work that teams do.

Supply Chain Management Ensures Quality In Open Source and Outsourced Components

Open sourced and outsourced components enable organizations to deliver high-quality applications faster, but they can't just use them and forget about them. Successful organizations put processes in place for selecting components and updating them as long as they use them.

Use DevOps And Supply Chain Principles To Automate Application Delivery Governance

Processes: The Modern Application Delivery Playbook

by Diego Lo Giudice
with Christopher Mines and Amy Homan
November 10, 2016

Table Of Contents

Notes & Resources

Forrester interviewed 36 vendor and user companies, including Accenture, Cast, CA Technologies, CloudBees, Consortium for IT Software Quality, Dell, Disney, Ford Motor Company, Headspring, Hewlett Packard Enterprise (HPE), IBM, IGM Financial, IT Revolution Press, JFrog, Mindspring, Mindtree, New York Life Insurance, Orbitz, PayPal, Perforce, Polarion Software, Rogue Wave Software, Scaled Agile Framework, Scrum.org, Siemens, Soasta, SonarSource, Sonatype, Sumo Logic, Tata Consultancy Services, Tenable Network Security, Travelers Indemnity, VMware, WestJet, and XebiaLabs.

Related Research Documents

Build The Right Software Better And Faster With Agile And DevOps Metrics

Improve Cybersecurity With DevOps

TechRadar™: Continuous Software Delivery, Q3 2016

FORRESTER®

Forrester Research, Inc., 60 Acorn Park Drive, Cambridge, MA 02140 USA
+1 617-613-6000 | Fax: +1 617-613-5000 | forrester.com

Traditional Governance: 19th-Century Solutions To Today's Problems

Traditional application delivery governance fails to account for the role and growth of automation in the application delivery process. Its focus on meetings, documentation, the creation of interim artifacts other than software, and manual reviews misses the point: You can extensively standardize and automate delivery processes — and that changes the nature of the governance challenge. Just as automated inspection replaced manual inspection of manufactured goods, automated development and operations (DevOps) practices including continuous delivery are replacing manual application delivery governance processes (see Figure 1).

FIGURE 1 Comparison Of Governance Practices Across Delivery Models

Measure	Traditional	Agile	DevOps
Success	Performance against planned schedule and budget	Working software, internally delivered	Working software, externally delivered
Compliance	Reviews of artifacts delivered	Log of demonstrations and retrospectives; reviews of artifacts	Data generated automatically from delivery pipeline
Cybersecurity	Standards compliance reviews	Standards compliance reviews; security epic and story demonstrations	Executable security tests, code scans; secure infrastructure via standard and automated environment provisioning
Architecture	Standards compliance reviews	Architectural epic and story demonstrations	Executable architecture tests, code scans

Manual Governance Is Slow, Costly, And Error-Prone

AD&D leaders speeding up the pace of software delivery face a choice: Change the way they govern or risk business failure. The good news is that the old way of governing wasn't working anyway — it didn't prevent project failures or security breaches, let alone improve the probability of success. The roots of the problem lie in assumptions that manual reviews of budgets, schedules, and activities provide any information about the likelihood of success or failure. Delivery organizations have learned the hard way that:

› **Misguided separation of duties and concerns prevents automation.** Business controls require the separation of certain kinds of tasks. Automation separates duties while eliminating manual steps; for example, you can substitute automated quality check decisions for subjective manual

release decisions. And automation simplifies other tasks: You can use static code analysis in place of manual code reviews for simple changes and randomize the selection of peers when you need manual code reviews.[1]

- **Manual processes can only catch so much.** @iamdevloper tweeted this perspective on code reviews: "10 lines of code = 10 issues. 500 lines of code = looks fine."[2] There simply are not enough people with enough time for manual processes to adequately oversee application delivery. Governance processes that depend on manual inspection are guaranteed to fail.
- **Traditional approaches arbitrarily measure activity, not outcomes.** Schedules and budgets reflect *guesses* about what it will take to deliver a solution. When reality differs, it's not necessarily a bad thing. The people making these guesses are often not involved in the actual delivery work. Governing based on tracking actual performance against these guesses provides very little insight into whether things are going right or wrong. The problem is not that organizations don't need estimates — it's that organizations enshrine estimates when they should not.

"Our annual budgeting process looked like this: All of the IT managers were gathered together to review the business's requests. Based on a few sentences of description, we were expected to come up with cost and schedule estimates. There was so much speculation involved that the estimates were almost worthless — except we were later held accountable for them." (Senior manager, financial services company)

- **Manually prepared scorecards provide ceremony with little substance.** People want to please; they don't like to look bad. Given the opportunity to interpret, they are biased toward the positive. Organizational cultures that discourage bad news by always stressing the positive can influence the information that people share.

Schedules and budgets reflect guesses about what it will take to deliver a solution. When reality differs, it's not necessarily a bad thing.

"We had monthly review meetings with management, requiring hours of slide preparation. The charts showed project status across a variety of categories. The colors tended to be mostly green and yellow, even on programs that insiders knew to be in deep trouble. The meetings and chart preparation were a complete waste of time. They told people what they wanted to hear, not what needed to be said." (Program manager, large multinational corporation)

- **Real feedback comes too late to be effective.** Despite intensive oversight, projects can go horribly wrong even when they appear to be on track for schedule and budget milestones. Stakeholders are not real users, and despite their sign-off or even involvement, users rebel when solutions don't meet their needs. And when decisions come late in the release cycle, such as when a late-cycle security review uncovers fundamental application problems, the fix is far more expensive than if it had been caught when the problem was introduced.

"We spent over a million dollars to build an application that the users rejected. The requirements were documented and signed off, and the project met all its milestones. The problem was that the requirements came from people who were not real users, and their idea of the right solution did not match reality." (AD&D leader, financial organization)

Fragmented And Dispersed Governance Processes Don't Help

The problem is not just that governance is too slow; there's also too much of it. In traditional software organizations, every silo has its own governance process: Operations has change advisory board meetings, enterprise architecture has review processes, and audit and security have their own processes. The result is compliance overload, where things still fall between the cracks.

- **Fragmented processes create duplicate work and confusion.** Release decisions require ensuring that environments match standards; so do security assessments, and so do architecture reviews. Even when the standards are consistent and up to date, it can be a lot of work. The more separate the processes, the more likely that they will be inconsistent. It's no wonder that developers simply give up, ignore standards, or violate standards and then ask forgiveness.
- **Funding and financing models force organizations to govern the wrong things.** Continuous delivery presumes a product model in which releases flow one after another, but funding models are often based on projects with a defined beginning and end. Finance organizations can create a mismatch when they demand plans and budgets that don't match the work effort.[3]

"Funding models were a huge headache for us as we adopted Agile practices. Finance wanted detailed plans and milestones for each release that didn't match the way we needed to work. The need to split work between capital and operating expenses was also a constant source of tension." (Program manager, large manufacturer)

Agile Practices Are Important And Necessary, But Not Sufficient

Agile software development practices improve transparency and visibility by openly sharing what developers are working on, what they have completed, and what they'll work on in the future. Burndown charts provide visibility into teams' productivity, and sprint-end demonstrations of working software make their progress concrete. Some organizations go further by incorporating peer reviews and pair programming to socialize code quality improvements. So far, so good. But many Agile transformations go wrong when they mistakenly ignore other governance practices, such as security processes, code analyses, and quality assurance (QA). These practices often remain largely the same, and compliance activities remain largely manual. The tension between speed and quality remains.[4]

Improving Visibility And Transparency Simplifies Governance . . .

Traditional governance relies on status reports and meetings to understand risk and progress measures. Agile practices and supporting tools like Atlassian's Jira Agile, CA Technologies' Agile Central, HPE's Agile Manager, IBM's Rational Team Concert, Microsoft's Team Foundation Server and Visual Studio Team Services, and VersionOne enable organizations to understand progress and status anytime, anywhere without the wasted time and effort of manual reporting. By applying Agile practices at scale, organizations have learned that:

- **Agile work management practices increase transparency and control.** Using Agile backlog and Lean Kanban techniques lets everyone see priorities and work status. Putting the business in charge of setting priorities connects tactical execution with strategic direction and establishes approval control without adding extra work.
- **Progress and productivity measures based on working software are more accurate.** In Forrester's most recent Agile software survey, 72% of respondents identified greater alignment with business as the top benefit realized from adopting Agile practices.[5] The survey also showed that the top metrics used were user stories (57%), features completed (45%), velocity (56%), and business value (49%). Collectively, these progress and productivity metrics increase transparency and confirm that teams are delivering the right software. Stakeholders are able to make corrections when they see something that doesn't seem right. Communication and trust improve — along with results.
- **Practices for enterprise-scale Agile improve the scope of transparency.** Originally, Agile practices focused only on the work of individual teams. Many organizations are successfully applying practices like the Scaled Agile Framework; techniques for scaling the Agile Scrum approach, like Nexus and Water-Agile-Fall, help companies use Agile techniques for not only products and projects, but also programs and portfolios.[6]
- **Measuring unplanned work improves visibility into where things are going wrong.** Agile's frequent planning and retrospectives provide organizations with greater planning granularity than the traditional approach. One side benefit of this is that unplanned work is easier to see, which gives organizations better understanding of where time was actually spent. Unplanned work gives insight into poor quality that causes unexpected rework, failure to protect and dedicate resources, and poorly understood hidden process steps.

"Measuring unplanned work helps organizations understand where they have lags, bottlenecks, and quality problems." (Brian Hardwick, managing director, Accenture)

. . . But Agile Management Practices Alone Don't Improve Speed And Accuracy

Agile product management tools improve reporting and visibility, but they don't improve other essential activities in the delivery pipeline, like building, testing, and deploying software. Agile practices are a tremendous improvement over traditional planning and teaming practices, but once these practices

improve, the bottlenecks shift to other activities. While Agile is an important, even essential, step in the journey to faster delivery with higher quality, many other steps remain. Organizations that have implemented Agile practices have learned that:

- **Stakeholder reviews don't substitute for real customer and user feedback.** While their opinions are important, stakeholders are neither the real users of the app nor the real customers of the business. They can be wrong, and relying on their feedback does not always lead to better results, even when that feedback is fast and frequent. Getting feedback from real users and customers is essential to really changing results.[7]
- **QA struggles to keep pace with Agile velocity improvements.** Agile practices can improve quality — but only if testing keeps pace. When manual QA can't keep up, Agile teams struggle. QA teams need deep transformation to support Agile and achieve speed with quality. Debunking old theories about testing is hard and requires AD&D and QA leaders to radically transform how they organize, how they test, and what they automate. In Forrester's Q2 2015 Global Agile Software Application Development Online Survey, 36% of respondents stated that they lacked proper Agile testing skills; 36% also claimed that lack of automation tools and/or infrastructure was an impediment to further Agile adoption.[8]
- **Agile largely ignores operations, and operations largely ignores Agile.** Agile focuses on what is potentially releasable, but significant problems can lie hidden until you actually deploy the application and it's running in production. Measuring deployment failures gives organizations a fuller picture of application health. When companies use Agile only for development and testing, and operations teams ignore it, organizations won't get the true value of Agile because software won't reach production fast enough. You've promised your business product owners faster delivery but are failing to actually deliver what you promised.

> "We found that tracking deployment problems gave us a more accurate picture of application health. We got a better sense for breakdowns in communications between development, QA, and ops, and deployment failures were a better predictor of the overall quality of the application." (Application development leader, large financial institution)

DevOps Practices Scale Agile And Improve Governance

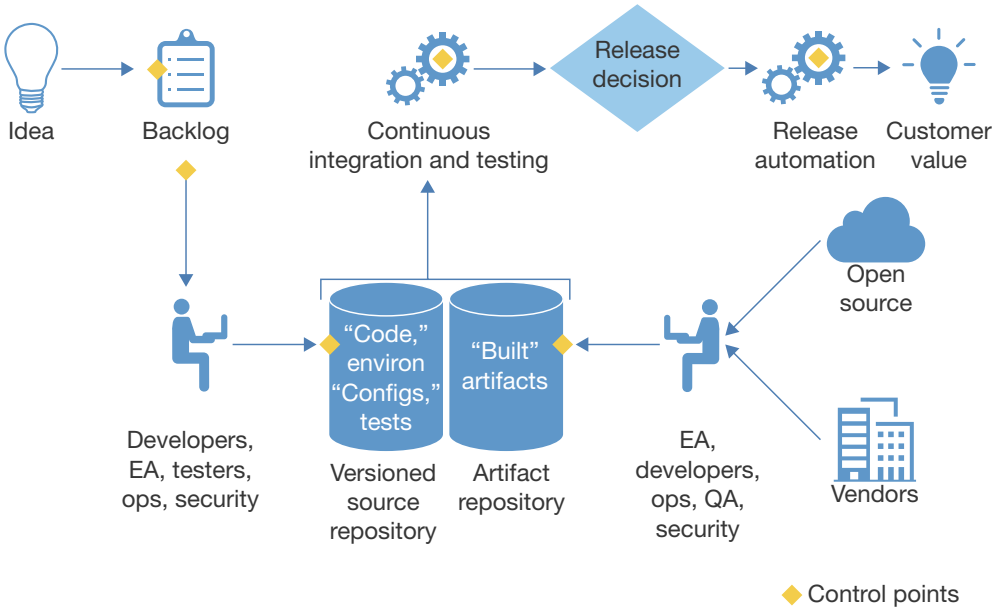
Organizations that think application delivery speed comes at the expense of quality are behind the times. DevOps turbocharges Agile by adding practices that automate application delivery pipelines while providing greater visibility and more points of control, not fewer. Traditional governance practices rely on manual controls because most of the processes are manual. Automating processes eliminates variability, reduces cost, and makes the remaining manual processes more visible (see Figure 2). Automation also solves the separation-of-duties problem without creating overhead; by subjecting

delivery pipeline automation to the same peer review processes as other code, duties remain separated with greatly increased speed and control. AD&D leaders who have implemented Agile and DevOps practices find the payoff in one or more of five areas:

1. Improved control over source code delivers better quality and security.
2. Continuous integration enables automated governance.
3. Automated environment management improves compliance.
4. Supply chain principles help govern the use of open source software.
5. Analytics is the truest measure of governance.

> DevOps practices enable highly automated application delivery pipelines that provide greater visibility and more points of control, not fewer.

FIGURE 2 Control Points In The Continuous Delivery Pipeline



Improved Control Over Source Code Delivers Better Quality And Security

As software's role in business has grown from supporting internal operations to providing differentiating customer experiences (CX), it has become a crucial organizational asset that firms need to protect.[9] At the same time, the cost of poor software quality has never been higher. Catching problems early prevents them from becoming embedded and systemic.

- **Securing source repositories prevents unauthorized access.** Software is an important asset. Not just anyone should have access to it, and only authorized personnel should be able to make changes. Controlling access to source code repositories is the first line of defense against security breaches and intrusions.[10]
- **Peer reviews find and fix problems at their source.** Peer code reviews put additional eyes on source code changes and are particularly important for areas of applications that perform sensitive processing such as financial transactions, safety-critical processing, performance-critical processing, and security-critical processing. Not every line of code can typically be peer-reviewed, so being selective and identifying critical areas is essential for applying peer review practices at scale. Open source tools like Gerrit and commercially supported tools like Atlassian's Crucible and Perforce provide support for peer code review practices.[11]
- **Static code analysis tools assess code quality when human eyes cannot.** Organizations use static analysis tools to give developers fast feedback on the quality of their code at the time they check it in, while the code is still fresh in the developer's mind and the problems are easier and less costly to fix. Feedback ranges from recommending good coding practices to early identification of serious security flaws. Benefits include reduced application failures in production, increased modularity, and increased maintainability. Popular static analysis tools include Cast, HPE's Fortify, Synopsis's Covetry, IBM's Security AppScan, Rogue Wave Software's Klocwork, and SonarSource's SonarQube.

"Developers should focus exclusively on code they are adding or changing, not on code which already exists. Otherwise, they are constantly introducing new problems, which, like a water leak that is never fixed, continue to cause more problems." (Olivier Gaudin, CEO, SonarSource)

Continuous Integration Enables Automated Governance

Continuous integration (CI) picks up when a developer commits code to a source code repository. Initially, it focused on building, integrating, and testing the changed code — but organizations have gradually expanded CI's use to cover all aspects of automated testing and verification, even going so far as using it to drive the entire delivery pipeline. Because of this, it enables a wide range of new points at which you can apply automated governance activities (see Figure 3).

- **Standardizing the delivery pipeline enables governance and lets developers create.** Creating a standard delivery pipeline as a service removes variability from the delivery process, creating checkpoints where you can apply visibility and control. But it also needs to be easy to use.

"Developers will only comply with governance policies if you make it really easy for them to do the right thing. If you put speed bumps and bureaucracy in their way, they will simply find ways to go around them. When that happens, we end up with unknown vulnerabilities — and that defeats the purpose of having a governance program." (Nigel Simpson, director of enterprise architecture, Disney)

Disney turned build automation into a shared service for developers. Rather than establishing their own continuous integration servers, teams use a ready-to-go enterprise service instead. The result was broad enterprise adoption through word of mouth rather than the kind of top-down mandate that developers frequently resist. In addition to build automation, this shared service provides code quality analysis, security scanning, and open source vulnerability review, which provide value to developers as well as enterprise risk management. It's a win-win for everyone.

› **Fast feedback improves quality by fixing problems close to their source.** Automating testing, including security testing and code quality analysis, as part of the CI process gives developers instant feedback on the code they are working on. They fix bugs faster and cheaper, leaving fewer issues for governance processes to worry about.

"When you get developers feedback as part of the CI process, they get that information while the code is still fresh in their minds. Developers are happier and more productive, and the code gets better." (David Mortman, former Chief Security Architect, Dell)

› **As testing coverage climbs, release decisions become no big deal.** Automating testing as part of CI also makes release decisions easier because it can enable organizations to achieve near 100% automated test coverage. Manual testing efforts are often too expensive or too short on time to achieve full coverage. CI- and API-driven testing provide the means to fix this. Automating all unit testing is the place to start. Some organizations take this a step further and don't allow developers to merge their code in the repository if it's not unit-tested.

› **Smaller releases, automation, and deployment alternatives simplify decisions.** Application release automation tools enable the simplification, standardization, and automation of release processes and provide the ability to roll back to a known viable version when needed, which makes the release process less stressful.[12] Architectural capabilities like feature toggling help simplify release decisions by decoupling release decisions from the decision to let users use a feature.[13] Blue-green and canary deployments limit the scope of releases.[14] Static analysis tools simplify release decisions by providing objective visibility into code quality. Finally, simply reducing the scope of a release reduces risk by limiting dependencies; less scope equals less risk.

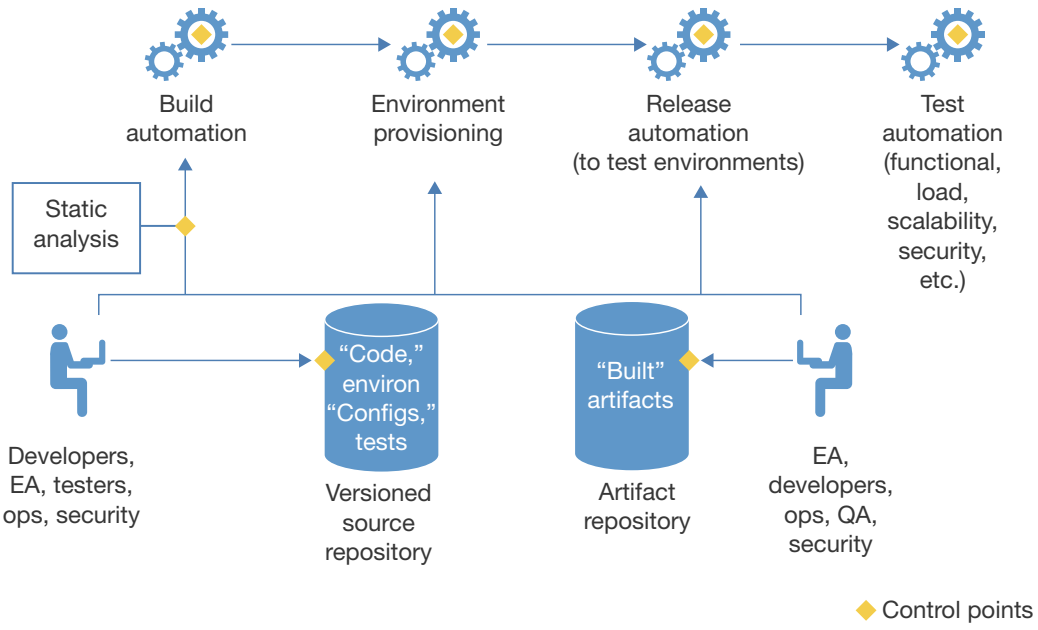
> When organizations manually manage environments, inconsistencies and errors creep in that cause costly downstream problems and security threats.

› **Visibility across the entire pipeline gives an easy and instant view of release health.** There are more than 17 categories of tools in the application delivery pipeline.[15] Organizations wanting a simple view of the state of a release must pull data from many sources to gain visibility. Taking the lead on creating a solution, Capital One launched an open source project, Hygieia, to provide a consolidated, comprehensive, and continuous delivery dashboard.[16]

› **Behavior-driven development (BDD) simplifies traceability and compliance reporting.** BDD, a form of test-driven development (TDD), means writing test specifications in a declarative language even before coding, in place of — or at least as an elaboration of — the requirements. TDD approaches let firms automate tests and trace them for compliance.[17] TDD and BDD practices are starting to pick up; according to Forrester's Q2 2015 Global Agile Software Application Development Online Survey, 24% and 14% of developers, respectively, frequently practice TDD and BDD.[18]

"Using TDD, we're able to automatically generate compliance documents from Cucumber [an open source TDD tool]. It makes regulatory reporting simple." (AD&D leader, financial institution)

FIGURE 3 Control Points In The Continuous Integration Subprocess



Automated Environment Management Improves Compliance And Eliminates Errors

Every aspect of application delivery requires environments: development, testing, and running the application in production. When organizations manually manage environments, inconsistencies and errors creep in that cause costly downstream problems and leave the organization open to security threats. Ensuring compliance in this manual world is slow, costly, and error-prone as well. Automating environment management enables organizations to:

- **Go faster while ensuring compliance with standards.** Organizations seeking faster application delivery face a conflict: Manual process control usually means limiting access, and limited access limits speed. Obtaining environments is often the first bump in the road. Standardizing environments and making them available on demand, automatically, both improves control and increases speed.

"Using the old process, it took weeks or months to get a test environment stood up. We had to fill out forms, get enterprise architecture to sign off, and then it went into an ops queue. Now we get standard configuration self-service without any delays. We can get on with testing, and enterprise architecture and operations can focus on more important things." (AD&D leader, multinational manufacturer)

- **Create an audit trail for all environment changes.** When organizations use infrastructure-as-code practices, they authorize, control, track, and version all environment changes, just like source code.[19] This creates a bill of materials for all components of an application, including operating systems, middleware, databases, and all configuration changes. Governance improves because of reduced variation and the ability to audit all changes.

"If we improve our component selection and traceability of components, we can significantly reduce the amount of unplanned, unscheduled work to improve developer productivity." (Brian Fox, CTO, Sonatype)

Managing The Open Source Supply Chain Improves Governance And Reduces Risk

Open source software (OSS) enables organizations to deliver high-quality applications to market faster. Using OSS and giving back to the supporting communities is so compelling that a major financial organization has adopted a policy requiring that anyone selecting commercial software solutions must first prove that there is no OSS that will solve the problem.[20] But open source also creates challenges for traditional governance processes. The complexity comes largely from its open nature; organizations are used to managing commercial vendor relationships, but with open source there is no contracting entity, no one to whom they can turn to fix things, and no retirement of old versions of software. To overcome these potential OSS problems, organizations are adopting new governance approaches:

- **Supply chain practices improve control over open source components.** The Sonatype 2016 State of the Software Supply Chain report provides some sobering statistics: There are more than 1.5 million unique components in the (Java) Central Repository alone; 72,000 have a known security vulnerability, and 324,000 have known restrictive licenses. More sobering, these numbers are up from the 2015 report![21] This is not an argument for not using open source, but for managing it better.

"Deming taught us that a supply chain process could maximize speed and quality by doing three essential things: maintain fewer and better relationships with suppliers, procure only the best components from those suppliers, and track the precise location of every component throughout the supply chain." (Brian Fox, CTO, Sonatype)

› **Assessing supplier quality extends to open source communities.** Commercial suppliers are relatively easy to evaluate; product evaluations and customer references are readily available. Similar information is available on open source projects from their source repositories.[22] Selecting components based on their usage and quality, including how frequently they are updated, helps to screen out components that leave applications at risk.

"Last year, the average enterprise downloaded 229,000 open source components. If properly sourced and managed, open source components are a tremendous source of energy for accelerating innovation. Our analysis of 25,000 applications demonstrated that 6.8% of components in use had a known security defect." (Brian Fox, CTO, Sonatype)

› **Using the highest-quality components means using the latest version.** Developers often use an open source component and then forget about it. Unless teams rebuild the software whenever a new component version is available, the application will be stuck with all the defects and vulnerabilities that existed at the time it was initially released. Tracking where components are used gives organizations visibility into risk exposure, and rebuilding applications when new component versions become available ensures that application quality and security improve over time. Artifact repositories like those from JFrog and Sonatype give organizations the means to improve the use of approved components by hosting them in governed, centralized locations for developers.

› **Static code analysis tools provide insight into code quality and risks.** Assessing code quality using static analysis gives organizations insight into code modularity, maintainability, overall quality, and security vulnerabilities.[23] The Consortium for IT Software Quality has defined a set of standards for software quality attributes that firms are adopting as a means for assessing the quality of components and deliverables in their software supply chains.[24]

Application And Operational Analytics Are The Truest Measures Of Governance

Ultimately, companies are most concerned with whether their application delivery efforts are winning, serving, and retaining customers and furthering their business technology agenda. Analytical data about customer usage and experience coupled with operational measures of performance, reliability, scalability, and security gives these organizations the means to measure success and improve future performance. Organizations use these metrics to:

› **Test hypotheses about what customers really need and deliver better outcomes.** Better CX information gives organizations insight into unmet customer needs and opportunities to improve those experiences, but the data is often buried in separate silos. Correlating information across application and operational measures deepens insights; a major retailer created data-infused journey maps from last year's anniversary sale data to improve CX.

"The key is to put all the data on the same timeline: latency, abandonment, lost revenue, performance, responsiveness, and availability. When you do that, you start to see what's really happening: What effect does 1 additional second of latency have on revenue? What does running an ad do for the close rate?" (Tom Lounibos, CEO, Soasta)

› **Improve customer experiences by reducing technical debt.** Operational analytics like response time and latency as well as production incident data provide organizations with early indicators of deeper technical problems. Comparing operational metrics over time gives organizations a way to spot problems early, before they become critical, and gives them the insights they need to prioritize CX improvements over new feature development.

› **Assess the privacy and security of company and customer information.** Customers expect privacy. When organizations violate that trust, brand loyalty evaporates in an instant. Monitoring environment drift, tracking where different versions of software components are installed so teams can easily patch them, and constantly monitoring activity enables organizations to avoid problems and fix them quickly when they occur.

Recommendations

Automate Governance To Increase Speed, Control, And Compliance

Increasing application delivery speed does not threaten governability and control if organizations do it right; governability is threatened by inconsistent processes and overreliance on manual controls. Simplifying and automating processes gives AD&D leaders greater control where it matters, while increasing their ability to respond to changing customer needs and competitor behavior. To improve responsiveness, compliance, and control, AD&D leaders need to:

› **Put everything under version control and manage access to repositories.** Versioning enables organizations to manage change and collaborate more effectively. "Everything" means not just source code, but also things like environment configurations, deployment processes, and externally sourced component binaries. Controlling access is more than just good practice; it gives organizations the means to protect their intellectual property against theft and unauthorized use.

› **Automate the delivery pipeline to improve speed, flexibility, visibility, and control.** Manual processes create inconsistency, confusion, and complexity. Creating a centralized, automated application delivery pipeline as a service and making it so attractive that every developer will want to use it is the key to improving governability at scale. Create unobtrusive control points in

the delivery pipeline. Don't prevent people from checking in code or building it and testing it, but prevent promotion to controlled environments. Continually improve the pipeline process by using no-blame retrospectives and measuring deployment pain.

- **Simplify and automate the software supply chain.** Use fewer and better suppliers. Use only the highest-quality components. Create a central repository of approved components of known quality, but establish a process that allows provisional additions of new components. Track what is used and where so that you can update applications with the latest components when better versions are available.
- **Continuously evaluate application quality.** Test software as soon as developers check it in, and don't allow developers to check in code unless it has at least unit tests. Build up to being able to completely test all functionality through APIs, and then expand to regression testing. Go beyond this to automate scalability, security, and performance testing. Use static analysis to evaluate code quality, supported by peer reviews. Leave manual testing to cover exploratory and user experience testing.
- **Correlate app, ops, and business metrics to improve CX.** Bring together disparate perspectives to provide a more complete picture of the customer's experience based on data. Use that data to form and test new hypotheses and new approaches to delivering superior customer and user experiences.

Supplemental Material

Survey Methodology

Forrester's Q2 2015 Global Agile Software Application Development Online Survey was fielded to 215 readers of Forrester's Agile reports with knowledge of their firm's Agile practices. For quality assurance, we screened respondents to ensure they met minimum standards in terms of content knowledge.

Forrester fielded the survey in April 2015. Respondent incentives included a complimentary copy of this report. Exact sample sizes are provided in this report on a question-by-question basis. This survey used a self-selected group of respondents and is therefore not random. This data is not guaranteed to be representative of the population, and, unless otherwise noted, statistical data is intended to be used for descriptive and not inferential purposes. While nonrandom, the survey is still a valuable tool for understanding where users are today and where the industry is headed.

Companies Interviewed For This Report

Accenture
Cast
CA Technologies
CloudBees
Consortium for IT Software Quality
Dell
Disney
Ford Motor Company
Headspring
HPE
IBM
IGM Financial
IT Revolution Press
JFrog
Mindspring
Mindtree
New York Life Insurance
Orbitz
PayPal
Perforce
Polarion Software
Rogue Wave Software
Scaled Agile Framework
Scrum.org
Siemens
Soasta
SonarSource
Sonatype
Sumo Logic
Tata Consultancy Services
Tenable Network Security
Travelers Indemnity
VMware
WestJet
XebiaLabs

Endnotes

[1] For a view on how the US Citizenship and Immigration Services achieved governance through automation, check out Mark Schwartz's recorded conference presentation. Source: "DOES14 - Mark Schwartz - U.S. Citizenship and Immigration Services," YouTube video, October 29, 2014 (https://www.youtube.com/watch?v=QwHVlJtqhal).

[2] Source: "I Am Developer," Twitter account (https://twitter.com/iamdevloper/status/397664295875805184).

[3] For more information on how application delivery, strategic planning, and portfolio management intersect, see the "BT Portfolio Management Best Practices Support Agile Delivery" Forrester report.

[4] Quality assurance teams need to debunk the tension between speed and quality and make testing a speed and quality enabler. Source: Mindtree, "Five must-dos for a continuous testing - Webinar featuring Forrester Research, hosted by Mindtree," Vimeo, August 12, 2015 (https://vimeo.com/136078353).

[5] Source: Forrester's Q2 2015 Global Agile Software Application Development Online Survey.

[6] More information about scaling Scrum practices is available at the Scrum website. Source: Scrum.org (https://www.scrum.org/Courses/Scaled-Professional-Scrum?gclid=CJOXypPP9McCFZU0aQod7hkLZg).

Information about the Scaled Agile Framework can be found at the Scaled Agile website. Source: Scaled Agile Framework (http://www.scaledagileframework.com).

For more information on Water-Agile-Fall, see the "Brief: Water-Agile-Fall Is A Steppingstone To Faster Delivery" Forrester report.

For additional perspective on how organizations are applying Agile practices to program and portfolio management, see the "Strategic Portfolio Management Is Agile" Forrester report.

[7] For a deeper discussion of the importance of real customer feedback and techniques for getting it faster, see the "Bring Design Practices To Application Development" Forrester report.

[8] For additional information on continuous testing best practices in an age of high-speed digital delivery, see the "Five Must-Do's For Testing Quality At Speed" Forrester report.

[9] For more detail on software-powered business, see the "The Software-Powered Business" Forrester report.

[10] For more information on how DevOps actually improves cybersecurity, see the "Improve Cybersecurity With DevOps" Forrester report.

[11] IT Revolution Press' DevOps Audit Defense Toolkit describes how peer code review practices can be run and documented to support IT audit concerns, including how randomly selecting peers for code reviews prevents collusion. Source: "The DevOps Audit Defense Toolkit," IT Revolution Press (http://itrevolution.com/devops-and-auditors-the-devops-audit-defense-toolkit/).

A list of source code peer review tools can be found at the DevZum website. Source: DevZum (http://devzum.com/2015/04/best-code-review-tools/).

Source code repositories that support workflows can be configured to support peer review processes as well. Source: Jason Cohen, "Peer Code Review with Perforce," Perforce (http://www.perforce.com/resources/presentations/user-conference-talks/peer-code-review-perforce).

[12] For a deeper dive into application release automation features and functionality, see the "Vendor Landscape: Application Release Automation Tools" Forrester report and see the "The Forrester Wave™: Application Release Automation, Q2 2015" Forrester report.

[13] Feature toggling is a technique for turning product capabilities on or off while the software is running. It decouples software deployment from activating capabilities for users. Source: Martin Fowler, "Feature Toggle," Martin Fowler, October 29, 2010 (http://martinfowler.com/bliki/FeatureToggle.html).

And there are pitfalls if the technique is not used with caution. Source: Jim Bird, "Feature Toggles are one of the worst kinds of Technical Debt," Building Real Software blog, August 6, 2014 (http://swreflections.blogspot.com/2014/08/feature-toggles-are-one-of-worst-kinds.html).

[14] Blue-green deployments are a way to release a new version of an application to one set of users while releasing another new version to a different set. Canary releases are similar, except that they selectively release a new version of the software to a subset of the users, leaving the remainder on the old release. Both techniques allow selective exposure of new capabilities to limit release risk. Source: "Canary release strategy vs. Blue/Green," Stack Exchange, May 19, 2014 (http://stackoverflow.com/questions/23746038/canary-release-strategy-vs-blue-green).

[15] For an analysis of the tools that comprise the automated application delivery pipeline, see the "TechRadar™: Continuous Software Delivery, Q3 2016" Forrester report.

[16] Source: GitHub (https://github.com/capitalone/Hygieia).

[17] Source: Cunningham & Cunningham (http://c2.com/cgi/wiki?TestDrivenDevelopment).

[18] For more information and data charts, see the "The 2015 State Of Agile Development" Forrester report.

[19] Source: Jafari Sitakange, "Infrastructure as Code: A Reason to Smile," ThoughtWorks, March 14, 2016 (https://www.thoughtworks.com/insights/blog/infrastructure-code-reason-smile).

[20] For more information on how open source is being used to improve application delivery practices, see the "Best Practices: Adopt Open Source Software To Improve Development Effectiveness" Forrester report.

[21] Source: "2016 State of the Software Supply Chain," Sonatype (http://www.sonatype.com/ssc2016) and "2015 State of the Software Supply Chain," Sonatype (https://www.sonatype.com/state-of-the-software-supply-chain).

[22] Examples include central.sonatype.org, npmjs.org, rubygems.org, and nugetgallery.org.

[23] A wide variety of static analysis tools are available. General-purpose tools include products like Cast Application Intelligence Platform, HPE Fortify, IBM AppScan, Parasoft, SonarQube, and Veracode.

[24] Source: CISQ (www.it-cisq.org).

FORRESTER®
CHALLENGE THINKING. LEAD CHANGE.

We work with business and technology leaders to develop customer-obsessed strategies that drive growth.

PRODUCTS AND SERVICES

- Core research and tools
- Data and analytics
- Peer collaboration
- Analyst engagement
- Consulting
- Events

Forrester's research and insights are tailored to your role and critical business initiatives.

ROLES WE SERVE

Marketing & Strategy Professionals

CMO
B2B Marketing
B2C Marketing
Customer Experience
Customer Insights
eBusiness & Channel Strategy

Technology Management Professionals

CIO
- Application Development & Delivery

Enterprise Architecture
Infrastructure & Operations
Security & Risk
Sourcing & Vendor Management

Technology Industry Professionals

Analyst Relations

CLIENT SUPPORT

For information on hard-copy or electronic reprints, please contact Client Support at +1 866-367-7378, +1 617-613-5730, or clientsupport@forrester.com. We offer quantity discounts and special pricing for academic and nonprofit institutions.

118681