DevOps is a quickly growing phenomenon that is quite literally transforming the way that software is developed, tested and supported in many businesses. Driving the transformation are a number of potential positive business outcomes. From an assurance, governance or security practitioner’s perspective, though, there are some additional aspects that need to be considered, planned for and thought through to ensure that the mission of the organization continues to be protected as these changes occur. This white paper outlines these considerations: the risk of DevOps (in adoption and nonadoption), controls that can help mitigate key risk areas, and specific actions that practitioners can take to ensure that the benefits of DevOps are realized while potential risk is mitigated.
INTRODUCTION

In early 2015, ISACA published an overview of DevOps, a relatively new development and operations methodology that many organizations are embracing as part of their technology programs and overall business strategy. That publication, titled *DevOps Overview*¹, provided a background on DevOps, pointing to specific business benefits that the model offers, outlining the success factors that make it work well, and describing many of the impacts and enterprise considerations for the use of DevOps.

DevOps can be very exciting, particularly from a software development and IT operations point of view. Specifically, by increasing the degree of integration between development and operations and leveraging automation to build on Agile software development methods (Agile), development of new functionality to support the business can move closer to a “continuous release” model. When applied strategically, this can mean more features in business-facing applications, developed in a faster time and with greater efficiencies throughout the development and release processes. This, in turn, can lead to reduced cost for the business as the overhead required to support and maintain business applications is decreased and increased business agility as new features can be developed more rapidly and with less “friction.”

Given the possible benefits, it should come as no surprise that many businesses are moving to embrace the concept. For assurance, security or governance practitioners, though, DevOps can be intimidating, even somewhat scary. Not only does it potentially impact the existing control environment, but it also can change what is already a complicated risk equation. These issues can be significant.

---

**DevOps can introduce new risk but mitigate other risk, from a technical and a business perspective. Looking at risk holistically means understanding both sides of that equation and making the right choice for each individual firm based on climate, risk tolerance, culture, project scope and other factors.**

---

IMPACT OF DEVOPS TO PRACTITIONERS

Before examining the specifics of the risk dynamics and suggested actions for practitioners in organizations adopting DevOps, it is important to understand the direct impact that DevOps can potentially have on practitioners. While many of the higher-level impacts to the organization as a whole are outlined in the “Impacts” section of DevOps Overview, it should be noted that these changes manifest themselves differently when it comes to a practitioner’s day-to-day activities.

Impactful Features of DevOps

To begin, it is useful to call out a few features of DevOps that are most directly impactful to practitioners. Obviously every environment is different, which means that specific impacts will vary. However, there are a few key features of DevOps that are useful to examine before looking in greater detail at how specific practitioner areas may need to adapt.

Increasing Pace of Release

First and foremost, much of the point of DevOps rests in reducing the cycle time required for release of applications. Agile moved development cycles from several months to shorter “sprints” (usually a few weeks); DevOps further reduces the amount of time involved in each “cycle.” Under DevOps, each cycle might be hours or minutes instead of weeks. In certain cases, a cycle may occur even more frequently. For example, John Jenkins, Amazon’s former lead engineer, went on record saying that Amazon pushes new code to production about every 12 seconds (11.7, to be exact) during the work week. Historically (i.e., prior to DevOps), a release cycle measured in seconds or minutes would be unheard of; now though, this pace is not only achievable but is, in many cases, the desired state. Note that scope and goals for projects and activities will impact cycle time (not every project will need to release as often as Amazon’s), but, in general, cycle time will be decreased.

From a practitioner point of view, this decrease in the cycle time for development can directly impact existing controls; it can introduce complexities in how current controls operate and, in some cases, may obviate certain controls entirely. An example might be a control such as software security or architecture review. Many shops utilize manual or automated techniques to evaluate application code for security vulnerabilities. Under a model like the Rational Unified Process (RUP) or Waterfall, there is often enough time in the release process for a review to occur with minimal disruption to the release process. This review could take the form of an automated code scan, formalized threat modeling approaches, or manual review of application code, components or architecture. The output of this review can provide direct feedback to developers, can initiate remediation activities (if security issues are severe enough) or engender further discussions such as compensating controls for high-risk issues. Figure 2 shows the ultimate goal of DevOps—to create a continuous delivery pipeline founded on continuous feedback, automation and efficiency.

Figure 2—Continuous Delivery Pipeline Approach

© 2015 ISACA. All rights reserved.
What happens, though, when the cycle time is minutes or seconds? Clearly, certain implementations of the control described above (e.g., manual code review) would become problematic. Not only is there less time available for the control itself to operate and complete, but further complications might occur if action is required as a result.

Automation
Automation is a significant facet of DevOps (DevOps Overview identifies automation as one of the critical success factors); in fact, numerous steps along the chain from development through to production might become automated under DevOps. The move to certain types of automation, depending on the specific development, release and operational processes that are being automated can impact the existing control environment.

From a development point of view, many aspects of development might become automated under DevOps: Tools such as Puppet and Chef might be employed to automate configuration management tasks; SaltStack or Octopus might be used to help in automating release orchestration activities; Gradle, Apache Ant™, Jenkins or Microsoft® Team Foundation Server™ (TFS) might be used to automate the build process. On the operations side, automation might be employed to keep track of application performance, application inventory, asset management or any number of other tasks. Even aspects of quality assurance (QA) might be fully or partially automated such as load testing, regression testing or functional testing of business logic.

This automation can greatly streamline the release process but, from a control standpoint, it can present some challenges. First, any process or control that requires human intervention to initiate will likely be affected if it cannot be incorporated into the areas of the process that are transitioning from manual to automated. Moreover, any control that introduces a significant delay in the process (e.g., because it is manual) will likely run counter to successful DevOps adoption. For example, a control such as a manual configuration checklist to validate that the production environment is appropriately secured may take hours to complete. If the business objective is to reduce the cycle time to minutes or less, a control that takes hours to complete might run counter to organizational goals and detract from the value that DevOps brings.

Consolidation/Integration of Duties
Lastly, keep in mind that one of the key reasons that DevOps is valuable is that it breaks down barriers between organizations in the software creation, development, release and support processes. Meaning, it removes the silos that have historically existed between groups such as QA, software architects, development teams, business teams, operations personnel, etc. Under DevOps, these teams work more closely together, interact more and may share responsibilities that may have historically rested solely within the “purview” of one group alone.

This matters from a practitioner point of view for a few reasons. First, shifts in responsibility can result in confusion: confusion about who “owns” particular aspects of the development and operations processes and confusion about the operation and maintenance of specific controls and countermeasures.

As boundaries of responsibility become “murkier,” it can be less clear from a security, assurance or governance standpoint who is the primary point of contact point for operation of specific controls and processes.

This can make it harder to audit those controls, make it harder to accurately ensure appropriate segregation of duties and make it more challenging to gather information to drive risk decisions.

Governance Impact
From a governance point of view, the impact of DevOps can be substantial. The decision to use DevOps (or not) is itself a risk decision. This white paper describes the risk aspects of DevOps in more detail in an upcoming section, but it bears noting here that DevOps can call into question risk decisions made in the past. If, for example, one or more legacy controls are impacted by DevOps, as described above, they may need to be revisited. If they are revisited, practitioners will need to ensure that the original intent of those controls is being met under the new paradigm. This can mean going “back to the well,” i.e., revisiting the factors that originally necessitated the implementation of that control in the first place and ensuring that the adapted control continues to meet the need.

There also can be an impact from a policy perspective because these policies are the artifacts that codify how value is created for stakeholders and how stewardship of technology resources is maintained. For example, it is possible that existing policies may need review to ensure that they still make sense in a DevOps context. In an ideal world, policy should reflect the overall organizational intent from a business goals standpoint (as compared to a technology goals), and these goals would be unlikely to change after DevOps is introduced. So, if a business goal of the organization is to have reliable applications that are resilient to attack, that goal would probably still be the case even
after the adoption of DevOps. However, in practice, the phrasing of specific statements in policy can sometimes be influenced by the techniques available to enforce it. Anything that is unenforceable or implies a particular development process that is not compatible with DevOps or constrains the ability of DevOps to be successful may require review and revision.

Performance measurement is a key element of IT governance.

A faster release cycle, in which many changes to application software occur very frequently, can outpace the mechanisms that organizations have in place to measure performance.

Analysis of performance information plays a role, too: If performance metrics cannot be collected and analyzed to ensure that goals are being met, governance decisions could be made with stale or inaccurate information. This can undermine the achievement of governance objectives, which requires consistent and reliable feedback to stakeholders throughout the organization to ensure that resources are best focused on goals.

Lastly, there is a potential impact on regulatory compliance that is possible should the organization not be careful to address those regulatory considerations during the move to DevOps. Keep in mind that certain regulations directly speak to software development and operations management tasks. For example, the Payment Card Industry Data Security Standard (PCI DSS) requirement 6.3.2 specifically mandates review of custom code for vulnerabilities using automated or manual processes while the sub-requirement of 6.4 outlines specific features of change control that should be implemented. In a Health Insurance Portability and Accountability Act (HIPAA) context, this is no less true: Standards and implementation specifications that speak to integrity (e.g., §164.312(c)(1)) or other technology-impacting goals can, if not planned around beforehand, be impacted depending on the nature of the DevOps processes put in place. It is not always a guarantee that development, operations or QA personnel will be focused on ensuring compliance with these requirements as they adjust their processes for maximum business flexibility and speed to market for new functionality.

Security Impact

DevOps can also have an impact from a security point of view. Application security controls—specifically those designed to locate or remediate application security vulnerabilities such as coding errors and architecture issues—are normally intertwined with the development process. Because that process changes when adopting DevOps, the controls operation can be impacted as a result.

During early phases of the development cycle, methodologies such as software architecture review or formalized threat modeling approaches may be affected. DevOps is, by design, an iterative process that may involve adjustment to component architecture or data flows in response to user needs and feature requests. In addition, application code scanning (i.e., static application security testing [SAST] and dynamic application security testing [DAST]) or manual code review can be impacted. Because these activities require application code to be extant to operate, the way in which they are employed may need to be modified to fit a DevOps approach. Moreover, output of review activities is valuable only to the extent that it feeds back into future development activities—for example, to the degree that they educate developers about security issues or drive mitigation and remediation activities. Because DevOps can in some cases mean a fully automated runway to production, how these results are captured and the way that they feed back into development processes may need to change as well.

Controls and tools designed to maintain the integrity of the configuration for production, development or intermediate environments (e.g., QA, staging, pre-production, integration environments) may also be impacted. Because DevOps approaches generally seek to apply automation to the configuration process (e.g., leveraging tools such as Puppet or Chef), security tools that validate configuration (e.g., vulnerability scanning tools) may need to be adjusted to accommodate DevOps. For example, running a monthly vulnerability scan may not prove particularly valuable when configuration changes can occur much more rapidly.

Because DevOps increases the alacrity by which software is released to production—and automates much of that release—the amount of time and “interaction points” required to mitigate security vulnerabilities and other issues is reduced. This, in itself, can decrease the utility of detective controls designed to find and report on application security issues and increase the utility of more life cycle-based application security approaches such as those that are designed to “build security in” to developed software. Examples of these approaches include those described in OpenSAMM (the Open Software Assurance Maturity Model), the BSIMM (the Building Security in Maturity Model), the Security Development Lifecycle, etc.

Assurance Impact

Assurance practitioners may also be impacted by DevOps. Probably the most significant area of impact relates to
separation of duties, which, under DevOps, can be fairly complicated. Historically, a developer with access to production is highly problematic; in fact, it is antithetical to having a managed, auditable environment. However, in a DevOps context, a developer may be able to initiate a deployment process that ultimately results (without the intervention of any other person) in a change to the production environment.

On the surface, that might sound like it compromises separation of duties and change control, but it is not necessarily so. It is very possible that all of the same checks and balances are still there, just in a different form. For example, the developer’s access may be severely restricted and tightly controlled and perhaps change control logs are created and tied to that change so that every adjustment is auditable.

The responsibility for approving code changes, creating a new build and implementing that build do not belong to the developer alone any more than they did before the adoption of DevOps. During the transition to DevOps, auditors can find themselves in a difficult situation. Because audit plans are usually assembled well in advance of the audit itself, it is possible that the plans may call for the testing of specific controls in a manner that may no longer be optimal in the fast-moving DevOps environment (either for the auditor or the auditees). An example is change management assessment for an application that has been migrated to use DevOps following an Agile change management process. Additionally, review activities that involve outside personnel (e.g., an audit or assessment driven via regulatory mandates or external oversight) may be impacted; they may need to be scheduled at appropriate times or they may require education of those external personnel about the processes in use.

It is productive to note at this point that, because DevOps relies so heavily on automation, that automated approaches to assurance can increase in value. Specifically, those organizations that have moved to a continuous auditing approach or are considering moving to a continuous auditing approach may find that integration of existing continuous auditing mechanisms into DevOps proves easier than for those using strictly manual methods.

There is risk associated with adopting DevOps and, of course, with not adopting DevOps.

### Risk of Non-adoption

Just as there is potential risk associated with adoption, there is risk that comes with nonadoption. Among the primary areas to consider when looking at the risk of nonadoption are the possible reduction in competitive advantage, the opportunity cost to a practitioner’s program and the shadow adoption that practitioners may fail to realize exists.

#### Business Risk

From a business point of view, a primary concern is the risk of decreased competitive advantage. An organization that is unable to leverage business value from DevOps is likely to be at a relative disadvantage if competing organizations are able to do so successfully. Examples of this phenomenon are best viewed from hindsight. For example, a hypothetical organization of 20 years ago that failed to adopt a technology like the personal computer or email would have probably seen its ability to compete drop off sharply as its competitors capitalized on the value of the technology.

Obviously, that is an extreme example that is unlikely to occur; most organizations, seeing that they are being outpaced, would reconsider the decision to adopt as soon as they noticed the erosion of their competitive advantage. But the example should illustrate the point: If there is a competitive advantage to be gained from a paradigm shift, trend or way of doing business and significant numbers of the organization’s competitors are gaining access to that advantage, the competitiveness of the organization in the marketplace is at risk.

#### Practitioner Opportunity Costs

In addition to the potential impact on competitiveness, there is a potential impact on practitioners as well. The same advantages

---

Making the best risk-driven decision for the organization involves looking at both options holistically and finding the optimal decision in light of risk tolerances and potential business benefits.

The following section examines new risk that can be introduced through the adoption and the nonadoption of DevOps and ways that DevOps can offset existing risk, i.e., risk that might already be there that employing DevOps could help mitigate. Ultimately, the decision whether to adopt DevOps must be enterprise-specific, based on the unique factors that vary from one organization to the next. However, examination of each of these areas of risk can help make that decision an informed one.
in timeliness, overhead reduction and agility that accrue to adoption in other areas can be of benefit to practitioners.

For example, the automation of system configuration may be useful from a security, assurance or governance standpoint. Practitioners can potentially advance their goals in each of these areas through this change:

- **Security**—Security-relevant configuration changes can be made quicker with use of automated configuration management. If, for example, a vulnerability scan or penetration testing activity reveals an unnecessary/undesirable network service (e.g., Finger, Echo, Telnet protocols), the configuration change required to support closing that service could utilize automation features to remediate the issue in the same manner as those made to support new application code.

- **Assurance**—Automated configuration systems often retain a record of when configuration changes are made, by whom and for what purpose. This information might be challenging to gather in an environment that is primarily driven by manual processes. Automated systems can be used to gather evidence about configuration to help streamline the audit process in a way that has reliability advantages over a manual approach.

- **Governance**—Collecting reliable metrics about processes is often facilitated by using automated approaches. These metrics can support the performance management aspects of governance activities. Likewise, policy enforcement goals can be advanced through the use of technical means to enforce those policies.

Other aspects of DevOps can also be used to support practitioner goals. If assurance and security activities can be automated and integrated into the release process, they can often add more value. For example, rather than manually initiating a code scanning tool at an arbitrary phase along the release pathway, automating that same scan when new code is pushed to production could offer significant advantages. Similarly, an automated vulnerability scan could be initiated automatically when configuration changes are made to the production environment. These are, of course, only a few examples of the value that adopting DevOps thoughtfully in coordination, collaboration and integration with other IT and assurance functions and processes might offer to practitioners to help progress their goals—value that is not available in an environment that adopts DevOps more narrowly or possibly only noncentrally.

### Shadow Adoption

Another risk associated with enterprise nonadoption is shadow adoption—adoption outside of the mainstream, corporate-sanctioned approach. In any larger organization, there can be areas where shadow use of any new technology, trend or paradigm can occur. This can arise as a result of deliberate noncompliance on the part of individual technical teams (they know they are deviating from the norm and do so anyway) or inadvertent noncompliance (they do not realize they are out of conformance with a set corporate standard or regulatory requirements). For example, budgetary considerations, lack of staff or other resource constraints might be reasons to perceive advantages offered under DevOps as a way to gain in efficiency and offset some of those challenges.

In an organization that has embraced DevOps centrally, there is an opportunity to standardize the selection and implementation of tools (including potentially those controls advocated by security and assurance teams). Where adoption occurs piecemeal, without central oversight and supervision, there is less standardization of processes and tools, and each instance of use is likely to require individual vetting and review from assurance and security teams to ensure that the use does not put the organization as a whole at risk.
RECOMMENDATIONS

There are a few things that practitioners can do now, as DevOps continues to gain traction. The most productive steps will vary depending on whether the organization has adopted DevOps centrally and formally.

Even in organizations that have not yet adopted DevOps formally, it should be on the radar of practitioners in the security, assurance or governance space. DevOps approaches can find their way into the organization rapidly (perhaps through shadow adoption or a merger and acquisition activity that suddenly makes use of DevOps as a required aspect of the new organization), so it is advantageous to develop an early understanding of its characteristics, risk trade-offs and potential utility. Practitioners’ advance preparation will enable them to know how to recognize DevOps when it arises and allow them to think through the risk dynamics (and their response to those dynamics based on their culture and risk tolerances) before the “pressure is on.” This will help them focus the attention of stakeholders in a constructive and systematic way.

There are also some advantageous steps that can be taken in organizations that have already adopted DevOps to a greater or lesser degree. First, it is important to establish a seat at the table, i.e., reach out to stakeholders in the development and operations communities to ensure that security, governance and assurance organizations are made aware of process changes as they occur.

Having a seat at the table ensures that practitioner areas are alerted to major process changes in advance.

It also means that they have contact points and relationships to discuss potential areas of utility from their point of view, e.g., to enable automated security or audit controls or determine what performance management information might be gleaned from the approach in use.

Once a dialog has been established, a constructive next step is to plan out the strategy for integration. Ideally, security, assurance and governance goals should be accounted for in the organization’s continuous release model. In some situations, this can mean automation of security and audit tools directly into the model, as described earlier. In situations where automation is not practicable, it may just mean collaboration with counterparts who are closest to the development process to ensure that existing controls currently in use do not become obviated. Additionally, this strategy should inform other planning activities already in process, e.g., an assurance practitioner developing next year’s audit plan may wish to consult with counterparts to ensure that the manner in which the audit is performed provides the information they want, tests the appropriate processes and controls, and makes sense in light of the new model.
## CONTROLS SELECTION

*Figure 3* is a table of controls that practitioners may wish to consider for their environments as the organization embraces DevOps. Not every control will fit every circumstance. Practitioners should evaluate carefully which controls make the most sense in light of their environment, their risk tolerances, their culture and other organization-specific factors. Moreover, the assessment steps listed are only suggestions; the manner by which the control is implemented may necessitate differing approaches.

<table>
<thead>
<tr>
<th>CONTROL</th>
<th>PURPOSE</th>
<th>IMPLEMENTATION</th>
<th>ASSESSMENT CRITERIA</th>
</tr>
</thead>
</table>
| **Automated software scanning**              | A fast release cycle can make it harder to review developed software for security or coding issues. An automated scan during the release process can look for these issues without interrupting the release path. | Implement via an automated dynamic or static scanning that triggers as part of the build, testing or release process. Severe issues may warrant further review while lower-priority issues (based on risk tolerance) might be flagged for mitigation in subsequent releases. | • Observe that application code scanning software is in place and kept current as new attacks are discovered.  
• Examine evidence such as log files to ensure that automated code scans are completed as part of release process. |
| **Automated vulnerability scanning**         | Tools such as Puppet and Chef provide automated configuration management functionality. Changes to configuration can impact the security of production platforms. An automated scan as part of the release process can locate those issues without introducing a bottleneck. | Implement via an automated vulnerability assessment that triggers as part of the release process. Severe issues may warrant further review while lower-priority issues (based on risk tolerance) might be flagged for mitigation in subsequent releases. | • Observe that vulnerability assessment software is in place and kept current as new attacks are discovered.  
• Examine evidence such as log files to ensure that automated vulnerability scans are completed as part of the release process. |
| **Web application firewall (WAF) or other layer 7 firewall** | In situations where application vulnerabilities occur that cannot be remediated quickly, a WAF or other layer 7 firewall (e.g., an extensible markup language (XML) firewall or Java virtual machine (VM) firewall) can provide a stopgap to mitigate the consequences while the underlying issue is remediated. | Implement via use of an in-line proxy filter (e.g., reverse proxy or web server filter) on the web server or in the communication path. | • Observe network architecture diagrams or other documentation to ensure that a WAF or other layer 7 firewall is in place.  
• Examine evidence such as log files to ensure that inbound requests are inspected by the WAF or layer 7 firewall. |
<table>
<thead>
<tr>
<th>CONTROL</th>
<th>PURPOSE</th>
<th>IMPLEMENTATION</th>
<th>ASSESSMENT CRITERIA</th>
</tr>
</thead>
</table>
| Developer application security training | Because the path between software development and production is streamlined and automated, training can help developers avoid the inadvertent introduction of vulnerabilities. | Train developers on secure coding techniques and commonly occurring application vulnerabilities such as the Open Web Application Security Project (OWASP) Top Ten. | • Observe the record of attendance or participation in developer-focused application security training.  
• Observe training materials to ensure that commonly occurring software vulnerability issues are covered.  
• Perform a periodic review of software to ensure that developers adhere to recommendations such as OWASP recommendations. |
| Software dependency management | Moving to a faster pace of release and automation of build and other intermediate release processes can sometimes make it easier for developers to introduce new dependencies—e.g., new open source or other supporting libraries, new supporting components and middleware or other dependencies. Implementing a process to track these can help offset issues should security or other issues impact supporting components and libraries. | Implement tools and/or processes to inventory, track and/or otherwise manage supporting libraries and underlying application components that might be newly introduced. | • Observe that a process is in place to offset unexpected dependencies.  
• Validate that a record of dependencies exists and newly introduced dependencies can be identified. |
| Access and activity logging | Separation of duties under DevOps can be fully realized by automated means (in fact, in some cases with more assurance). However, this depends on logging being enabled and logs being retained. | Implement via logging of access and developer activity that results in changes to production code. Logs should contain, at a minimum, the individual responsible for changes and the time that those changes were made. | • Observe log files to ensure that logging is enabled.  
• For a sample of production changes, observe that the change can be mapped back to specific developers.  
• Review a sample of historical production changes to ensure that log files are retained. |
<table>
<thead>
<tr>
<th>CONTROL</th>
<th>PURPOSE</th>
<th>IMPLEMENTATION</th>
<th>ASSESSMENT CRITERIA</th>
</tr>
</thead>
<tbody>
<tr>
<td>Documented policies and procedures</td>
<td>DevOps processes, while automated and alacritious, should still employ rigor and discipline to ensure that security and risk management goals are met. Having documented policies and procedures describing the release flow is advantageous.</td>
<td>Develop policies and procedures that outline the development and release life cycle. Include policies and procedures in developer training programs.</td>
<td>Review policies to ensure that they exist and cover all aspects of the production release process.</td>
</tr>
</tbody>
</table>
| Application performance management (APM) | As development and operations processes become more fluid, it is important that applications continue to perform as expected and remain available to stakeholders. Application performance management tools can help both provide metrics about application performance and flag potential problem areas when/if they occur. | Establish a mechanism for tracking application performance and availability. This can encompass APM-specific tools (such as commercial or open source APM products) in conjunction with processes that leverage those tools to collect metrics and drive operations tasks. | • Review operations tools to ensure that application performance and availability issues can be identified.  
• Review documented processes and procedures to ensure that appropriate personnel are notified or appropriate activities are conducted in light of an issue. |
| Asset management and inventorying | As DevOps accelerates the development process, consider implementing automated or manual methods to retain a record of applications and important information about them such as:  
• Business owner and purpose  
• Domain and subject matter experts  
• Physical or virtual location  
• Supporting controls and countermeasures | Utilize either an automated tool or manual processes to maintain a record of applications and supporting information about them. Establish tools and/or processes to ensure that this information is kept current and updated. | • Observe tools and/or processes used in support of the application asset management goals to ensure that they are operational.  
• Review the master inventory for accuracy; e.g., review a sample of applications on the inventory to ensure that entries are accurate and complete. |
Continuous auditing and/or continuous monitoring

<table>
<thead>
<tr>
<th>CONTROL</th>
<th>PURPOSE</th>
<th>IMPLEMENTATION</th>
<th>ASSESSMENT CRITERIA</th>
</tr>
</thead>
</table>
| Continuous auditing and/or continuous monitoring | Moving to a more real-time and ongoing validation of controls can enable the organization to ensure that controls continue proper operation and that countermeasures are performing as expected. | Establish a process and supporting tools to continuously validate proper operation of controls. | • Observe mechanisms used to collect information about the control operation.  
• Validate that coverage is sufficient to address all applications and environments in scope.  
• Observe a record of data collected to ensure that output is complete and accurate. |
CONCLUSION

There appears to be no question than an increased level of integration between development and operations, coupled with the use of automation to build on Agile software development methods, can move an enterprise closer to a “continuous release” model for new applications and functionality. When applied strategically, the results can be impressive, running the gamut from reduced costs to increased agility.

While those benefits are real, they do not come without a price.

The very features that make DevOps attractive to businesses can cause concern for assurance, security or governance practitioners.

The changed environment that accompanies the adoption of DevOps can potentially impact the existing, carefully crafted control environment and the enterprise’s accepted level of risk.

As with most new technologies, there is risk associated both with adopting DevOps or choosing not to adopt it. In enterprises that adopt it, the reduced cycle time required for release of applications and the increased use of automation can directly impact existing controls by introducing complexities in how current controls operate and, in some cases, obviating certain controls entirely. The governance decisions relating to risk, including decisions made in the past, may require rethinking, and performance metrics on which business decisions are based may need adjustment. Many security controls, which are often intertwined with the development process, can be compromised. And assurance practitioners will have to address a particularly significant area of impact: separation of duties.

The risk of nonadoption is no less daunting. Enterprises that elect to decline this new technology face a potential loss of competitive advantage as other businesses leverage the benefits that DevOps provides. Shadow adoption can be an issue as well when certain departments within the business “go rogue” and adopt DevOps on an informal and unofficial basis, thereby bypassing centralized oversight and control.

There are productive steps that governance, security and assurance professionals can, and should, take now to ensure a seamless transition if the decision to adopt is made.

Much benefit can be gained by starting the dialog within the enterprise in advance to raise awareness of both the benefits and potential risk of DevOps and the related importance of controls.

After adoption, there are numerous controls that can be undertaken, several of which are outlined in this white paper.

The important thing is to avoid doing nothing. It is likely that every reader of this white paper—if not in his/her current company, but in the next one—will engage with DevOps in some way. It is better to be prepared for the apparent inevitability.
ACKNOWLEDGMENTS

ISACA wishes to recognize:

**Author**
Edward Moyle
ISACA, USA

**Expert Reviewers**
Ms. Sai K Honig
CISA, CIA, Australia
Ricci Ieong
Ph.D, CISA, CCSK, CEH, CISSP, eWalker Consulting (HK) Ltd., Hong Kong
Clay Moegenburg
CISA, CISM, CRISC, CGEIT, CIA, ACP PCI, PMP, Synectus Group LLC, USA
Matthew Mutee Mutiso
M-KOPA, Kenya
Ronke Oyemade
CISA, CRISC, PMP, Strategic Global Consulting LLC, USA
Andriy Rybalchenko
CISA, CISM, EastOne, Ukraine
Emil David Skrdla
CISA, CISM, CRISC, CEGIET, CISP, ITIL v3, PCI ISA, PCIP, University of Oklahoma, USA
Eva Sweet
CISA, CISM, ISACA, USA

**ISACA Board of Directors**
Christos K. Dimitriadis
Ph.D., CISA, CISM, CRISC, INTRALOT S.A., Greece, International President
Rosemary M. Amato
CISA, CMA, CPA, Deloitte, Amsterdam, The Netherlands, Vice President
Garry J. Barnes
CISA, CISM, CGEIT, CRISC, MACD, Vital Interacts, Australia, Vice President
Robert A. Clyde
CISM, Clyde Consulting LLC, USA, Vice President
Theresa Grafenstine
CISA, CGEIT, CRISC, CPA, CIA, CGAP, CGMA, US House of Representatives, USA, Vice President
Leonard Ong
CISA, CISM, CGEIT, CRISC, CPP, CFE, PMP, CIIP, CIPT, CISSP-ISSMP-ISSAP, CSSLP, CIBCM, GCIA, GCIP, GCISA, GCFA, ATD Solution, Singapore, Vice President
Andre Pitkowski
CGEIT, CRISC, OCTAVE, CRMA, ISO27kLA, ISO31kLA, APT Consultoria de Informatica Ltd., Brazil, Vice President
Eddie Schwartz
CISA, CISM, CISSP-ISSEP, PMP, WhiteOps, USA, Vice President

**Gregory T. Grocholski**
CISA, SABIC, Saudi Arabia, Past International President
**Tony Hayes**
CGEIT, AFCHSE, CHE, FACS, FCPA, FIIA, Queensland Government, Australia, Past International President
**Robert E Stroud**
CGEIT, CRISC, USA, Past International President
**Zubin Chagpar**
CISA, CISM, PMP, Amazon Web Services, UK, Director
**Matt Loeb**
CAE, ISACA, USA, Director
**Rajaramiyer Venketaramani Raghu**
CISA, CRISC, Versatlist Consulting India, Pvt., Ltd., India, Director
**Jo Stewart-Rattray**
CISA, CISM, CGEIT, CRISC, FACS OP, BRM Holdich, Australia, Director