PARASOFT®

Why DevOps Needs Static Analysis

DevOps is a collection of practices that facilitate the cross-departmental collaboration and communication necessary to help organizations optimize and accelerate their development processes. Its roots can be traced to the rise of iterative development methodologies that require a different way of operating that blends software development, IT, and QA. DevOps carries with it the potential to remove roadblocks that prevent organizations from meeting modern business challenges.

DevOps also challenges us to reevaluate the role of some software quality practices, such as static analysis, within the development process. To be clear, there are no "DevOps tools" that you can purchase and deploy that will suddenly enable you to start "doing DevOps", but there are necessary mechanisms for implementing a DevOps strategy. In this paper, we'll discuss how static code analysis enables organizations to start doing DevOps.

Automated Feedback Loop

The DevOps movement facilitates a partnership between the departments that have an impact on the implementation of requirements. By sharing knowledge and tasks across departments, organizations create an efficient process for accelerating the SDLC while improving quality processes. For DevOps to be effective, however, an automated feedback loop must be implemented that enables the consistent application of quality policies as requirements progress from creation to production.

Most organizations have a feedback loop in place for implementing requirements. At the most basic level, the feedback loop begins with policy and progresses through implementation, measurement, post-mortem, policy adjustment, and re-implementation. But in a traditionally operating business, the loop is a cumbersome process executed by siloed departments that often lack adequate knowledge of how upstream or downstream teams operate.

Automation is critical. It is far too costly from a resource perspective for people to stop their normal development and testing tasks to gather data related to the feedback process. Manually producing the feedback loop also risks introducing inconsistent and inaccurate analytics that hamper an organization's ability to improve development processes and, ultimately, software quality. Organizations can automate the feedback loop by implementing a central point of integration for all the components associated with the development and testing infrastructure. A centralized platform enables BTSs, RMSs, source control, analysis tools, and other components to collect artifacts so that DevOps teams have access to reliable, consistent information that helps them not only prevent defects, but accelerate the entire SDLC.

Preventive Phase of the SDLC

As the implementation of a requirement progresses, the feedback loop helps developers prevent errors and understand the impact of change as the requirement matures. In terms of doing DevOps, the feedback loop enables the bidirectional flow of analytics from development to deployment and vice versa. Operations contribute to the collaborative process in the form of shaping some aspects of development policy, while development enables the policy to prevent defects that affect operations. This preventative phase of the SDLC is the heavy-lifting process most organizations are familiar with in terms of static code analysis:

1. Determine which static analysis rules to run per the organization's development policy (also informed by feedback from operations)
2. Run the analysis and address any defects according to policy
3. Repeat until the analysis returns an acceptable number of violations

What's especially important about this phase from a DevOps perspective is that it involves the ability to capture, measure, and analyze the expediency of the activities performed. Organizations must understand which functions of their quality processes are helping the organization achieve its goals—and which functions are just making noise. This leads into the second phase of development testing for DevOps: the application of a *post-analysis* analysis to the development policy.

The findings exposed during the preventative phase must be measured, prioritized, and fed back into the development process in the form of an iterated policy. This helps teams that are downstream in terms of implementing the requirement, such as QA and IT, not only do their jobs, but contribute to a continuous process that results in safe, reliable, and responsive code.

Process Intelligence for DevOps

A development testing platform enables the collection and correlation of artifacts associated with the development process. This infrastructure, which automates the feedback loop that enables post-analysis analysis, is the minimum requirement for doing DevOps. A highly-integrated development testing platform extends this concept by enabling the organization to overlay additional data sets that help the organization not only address software quality, but also pinpoint risks and highlight areas for improving the development process.

This process intelligence layer significantly advances post-analysis analysis by exposing potentially risky data patterns collected during static analysis and other software quality activities. Findings are organized and prioritized into actionable information. From a DevOps perspective, process intelligence is the brain of the development testing platform that constitutes the feedback loop. By implementing a development testing platform that includes integrated software quality tools, organizations turn existing development infrastructure into an environment optimized for DevOps.

Static Analysis for DevOps

Data must feed the automated feedback loop for an organization to realize the benefits of DevOps. Along with unit and regression testing, automated static analysis creates the data necessary for imbuing the feedback loop with information analyzed during post-analysis analysis. From this perspective, static code analysis serves a few roles:

- Ensures code quality during the preventative phase
- Provides the big data required to improve the development process
- Facilitates the machinations of the DevOps automated feedback loop

Organizations can implement static analysis in myriad ways, but for the purpose of static analysis for DevOps, we'll discuss two types as distinguished by their purpose: 'finding bugs' before release, and

hardening code. The bug-finding flavor is the minimum type of static analysis an organization should run. But it is far more beneficial from a process-improvement perspective to examine the defects that are detected during release or QA to determine if there is a way to harden code and eliminate the possibility of these defects occurring again.

This is not to detract from the benefits of running static analysis with the intention of finding bugs. But consider that when used in this way, it is highly unlikely that testers are finding all instances of the defect. The preventative form of static analysis, on the other hand, involves enabling rules that specifically target systemic defects that create downstream bugs. For example, running dynamic analysis may simulate flow in an attempt to inject tainted data into the code, but by running specific preventative static code analysis rules, organizations can prevent the possibility of tainted data ever making it into the code.

Policy

The type of static analysis an organization deploys to implement its DevOps strategy depends on the organization's development policy. Not all organizations create mission- or safety-critical software and their static analysis strategy may not call for a hardening of the code. But as the DevOps feedback loop cycles through, organizations may adjust their policy in response to the post-analysis analysis data.

For example, if the policy states that finding and remediating a certain class of violations is acceptable during QA, but that the violations correlate to a more severe type of failure in the field, then the organization may decide to enable additional rules aimed at preventing the defect. When this process occurs within the context of a development testing platform capable of delivering process intelligence, the organization can apply additional metrics, such as code authorship, and fine tune their policies even further to, for example, prevent junior developers from accessing code associated with failures in the field. In this sense, policy both informs and is informed by static code analysis.

Conclusion

To be clear, DevOps can't be purchased and deployed. It is a *way* of building software that promotes tighter cross-departmental collaboration and communication. The goal is to move beyond the old ways of crossing our fingers and hoping that the application functions as expected when it's passed to the next department. This *has* to change. That said, there are tools and strategies to help facilitate the adoption of DevOps. Static code analysis, especially as part of a development testing platform, can help organizations achieve the promise of smoother processes and faster release cycles DevOps affords.

About Parasoft

Parasoft researches and develops software solutions that help organizations deliver defect-free software efficiently. To combat the risk of software failure while accelerating the SDLC, Parasoft offers a Development Testing Platform and Continuous Testing Platform. Parasoft's enterprise and embedded development solutions are the industry's most comprehensive—including static analysis, unit testing, requirements traceability, coverage analysis, API testing, dev/test environment management, service virtualization and more. The majority of Fortune 500 companies rely on Parasoft in order to produce top-quality software consistently and efficiently as they pursue agile, lean, DevOps, compliance, and safety-critical development initiatives.

Contacting Parasoft

Headquarters

101 E. Huntington Drive, 2nd Floor
Monrovia, CA 91016
Toll Free: (888) 305-0041
Tel: (626) 305-0041
Email: info@parasoft.com

Global Offices

Visit www.parasoft.com/contact for contacting Parasoft in EMEAI, APAC, and LATAM.

© 2016 Parasoft Corporation
All rights reserved. Parasoft and all Parasoft products and services listed within are trademarks or registered trademarks of Parasoft Corporation. All other products, services, and companies are trademarks, registered trademarks, or servicemarks of their respective holders in the US and/or other countries.