turnkey

Quality Driven DevOps
5 Steps to More Effective QA Automation in DevOps Environments

By Daniel Gannon

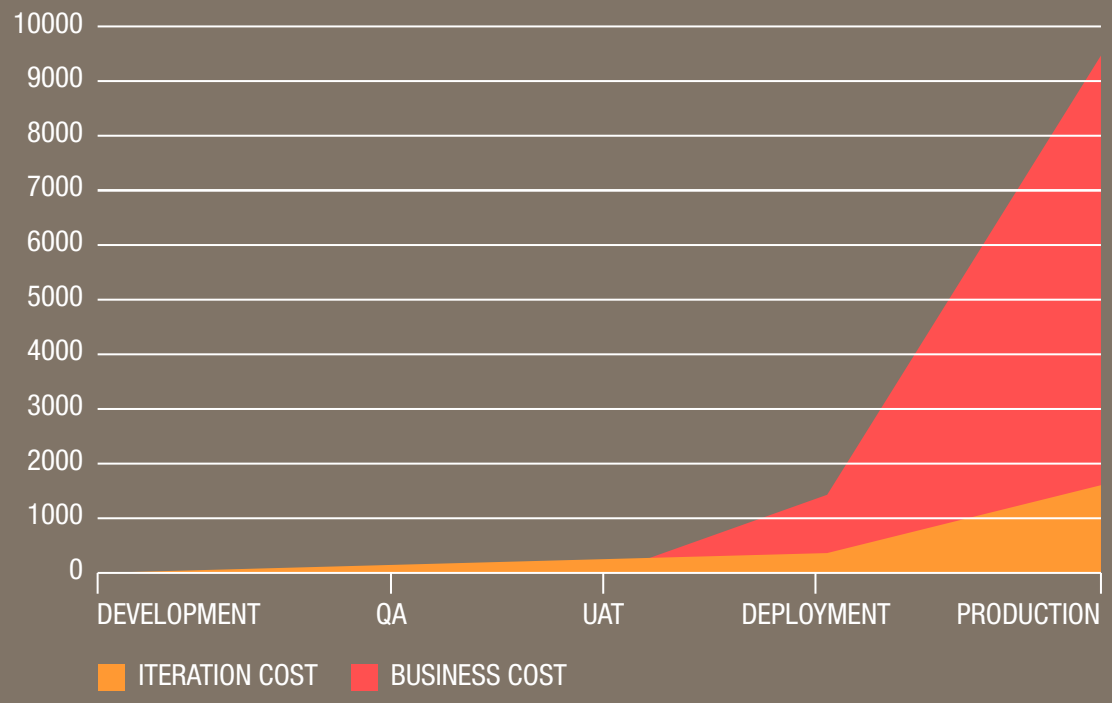
Allocating sufficient time and resources to thoroughly test enterprise applications is a continual challenge – especially in DevOps and Agile environments. Many organizations have not yet fully leveraged advances in Test Automation to realize the promises of DevOps and the benefits of faster time to value, increased agility, and better alignment with the needs of the market.

This paper presents a new strategy for improving QA and Delivery performance in today's rapidly evolving DevOps world, with a focus on 5 key areas of opportunity that are empowered by advancements in modern Test Automation software.

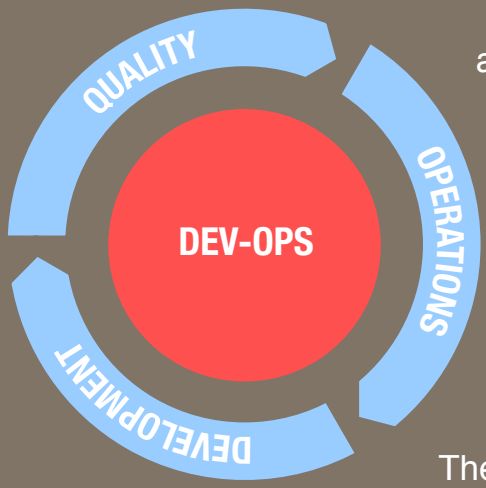
Executive Summary

Quality Assurance has been an integral part of DevOps from the very beginning. It all started in 2007 when Patrick DeBois led testing for a large data center migration and found that the conflicts between Development and Operations made his job of testing far more difficult than necessary.

Today just about everybody is either talking about or using DevOps. Not only do Horizon III companies like Amazon®, Facebook® and Netflix® embrace DevOps, but also many longstanding names like Fidelity®, Walmart®, and Sony®.[1] The benefits are well publicized, including decreased time to value, agility, and better alignment with customers' needs - regardless of whether the customer is an internal department or an organization's primary source of revenue. A lot has changed in the past eight years, but poor software quality still remains a serious issue. The good news is that the many advantages DevOps brings can have a lasting impact on your Quality Assurance initiatives.



DevOps and its predecessor Agile development provide a gateway to the best practice of "find and fix early." A defect found early minimizes costs - sometimes needing just a few hours of a developer's time. In contrast, traditional waterfall processes push for feature completion first, followed by lengthy QA testing and bug fixing cycles. As time persists the cost of an issue grows beyond development



and QA to also impact Business Analysts, Operations, and most importantly - end users. These "customers" – whether internal or external – face costs that are orders of magnitude higher than the time to find and fix problems earlier. Most significant is the ongoing "tax" that organizations pay in lost revenue, brand degradation, and delayed time to market due to deployed software defects.

The rapid and frequent iterations that are foundational to DevOps demand increased test velocity, flexibility, and agility. This casts an obvious spotlight on Test Automation, but all automation tools and methodologies are not created equal. Unfortunately, the majority of QA tools in production today were designed for yesterday's lengthy waterfall testing cycles and are much too inflexible, inaccessible, and slow to meet the needs of DevOps and Continuous Delivery.

5 Strategies for Improving DevOps and QA Alignment

This paper presents a new QA Strategy to support improved QA performance in today's rapidly evolving DevOps environments, with a focus on 5 key areas of opportunity.

1. Achieve Greater Collaboration
2. Shift Left – Integrate Quality with Development
3. Accelerate Application Delivery
4. Further Automate Testing
5. Getting Started with Test Automation

Strategy 1: Attain Greater Collaboration

Real-time collaboration between Development, QA, and Operations is an essential part of an effective DevOps strategy. DevOps success demands that the team's roles adapt and evolve. Gone are the days when developers and operations have no responsibility for the resulting application quality. QA teams need to move beyond running only well-established "happy-path" tests with approaches that support continuous process and quality improvement. Finally, application owners and Business Analysts must demand a more active role to ensure that tests cover the broader aspects of software that are key to the business.

DevOps also requires rapid communication between teams. Daily or weekly team scrums help, but daily and direct (practitioner to practitioner) information exchange, with management visibility, is also required. Software technology provides an effective conduit for communication, test review, and planning. Test Automation not only documents what tests are being performed, but also records the success or failure of each. In contrast, manual, ad-hoc testing and custom scripts are not only time consuming, but are also nondeterministic, producing a wide variance of results based on the person performing the tests.

Today's advanced Test Automation tools facilitate improved collaboration within DevOps organizations, while traditional QA "tools" were developed for QA silos. They not only restrict who can use the QA tools, but insert unneeded barriers to the shared visibility that enterprise software applications demand. The speed and agility demands of DevOps require a more collaborative QA approach to quality automation that are readily available to a wide range of teams and departments.

The antithesis of collaboration is finger pointing and blame. Does this scenario sound familiar? High visibility issues are uncovered post-release and QA teams bear significant blame. But they pass the buck to development, "who created the bug in the first place." Development's comeback is "it worked fine during development" and then pass the blame to operations based on the logic that it was not deployed correctly. During this whole mess, Business Analysts and application owners share the blame for not knowing what they want in the first place, producing incomplete requirements, or continually changing their minds.

Keys to better collaboration and eliminating finger-pointing include:

1. Eliminate "us versus them" from your vocabulary
2. Keep communication frequent, concise, and documented
3. Adopt a mentality of eliminating technical debt
4. Select software QA tools that can be used and understood by everyone

The source of bugs is not isolated to just development teams. Data, dependencies, and infrastructure changes can also be root causes. Full stack, functional, end-to-end business process testing has the advantage of testing what the end-user is actually going to experience. This not only aids QA teams, but also gives development and operations the ability to re-run tests and pinpoint the impact that changes in code, data, or IT infrastructure have on the success or failure of a battery of tests.

The collaborative focus of DevOps is a paradigm shift from the days where BAs develop requirements, developers code, QA tests, and the operations team deploys. The unilateral synergy that is a result of DevOps not only improves the work environment, it also increases agility, reduces time to market, and better aligns testing with business objectives.

Strategy 2: Shift Left – Integrate Quality Processes

To many, the interpretation of the term "Shift Left" implies the transfer of the burden of testing from a QA team to the development teams to ensure quality. The concern with this point of view is that the value added by QA professionals, application owners, Business Analysts, and others is lost. **Shift Left is the process of designing, planning, and implementing tests very early within a sprint, and keeping QA activity concurrent with the development process.**

The biggest risk to on-time deployment is the discovery and resolution of defects late in the process. This puts product management and application owners in the precarious position of having to deploy known defects in order to hit a deadline. Shift Left reduces the number of last minute surprises and when combined with the development best practice of "done is done," the result is transformational. No longer are defects the primary constraint to a timely deployment.

Unit and integration testing are well accepted within progressive development teams, while end-to-end functional testing has traditionally been reserved for the QA teams. The lack of meaningful, end-to-end testing within development is largely due to a failure to insert business owners into the DevOps process and the amount of heavy-lifting required to execute and automate these tests. This omission not only increases the likelihood that key issues go undetected until late in the cycle, but also impairs development agility by restricting their ability to make drastic changes (like refactoring) without fear of breaking something.

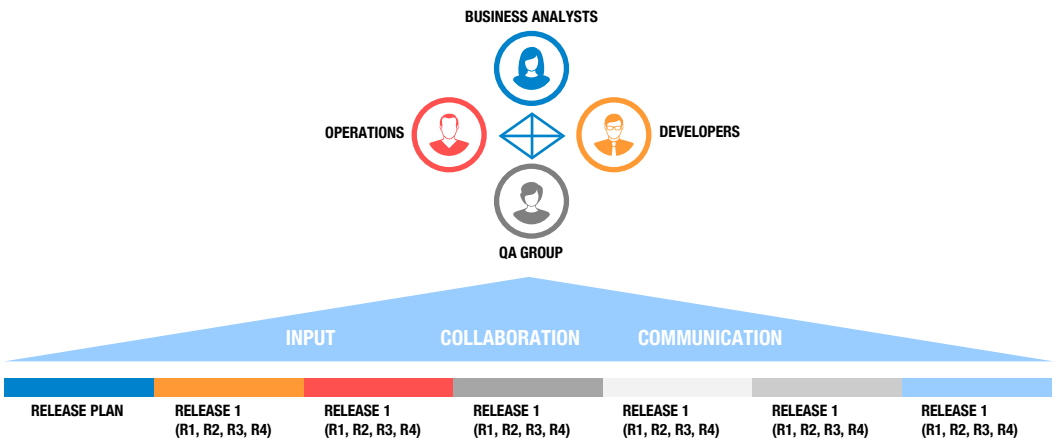
As author Stephen Vance highlights in his book, *Quality Code: Software Testing Principles and Practices*, "of all the practices you can leverage to assist your craftsmanship you will benefit most from testing."[2] By front-loading tests, developers have the benefit of fixing issues while the code and changes are still fresh in their minds.

Script based Testing in DevOps?

Script based testing works well for unit and integration tests where the developer of test scripts has a clear understanding of the code or modules under test. With functional testing those who understand the business cases need to be involved to construct effective tests, yet these people rarely have the programming skills to develop test scripts. In addition, DevOps elevates the need for test agility, while custom scripts tend to be difficult and time-consuming to change. Scriptless Test Automation software gives everyone the ability to create and modify tests and test data. This eliminates the disconnect between those best suited to develop the test cases (test engineers & business analysts) and skilled test automation developers required to code the automation. Scriptless test automation also improves visibility into the testing process for all stakeholders to deliver a successful outcome.

Having access to functional end-to-end business process tests puts developers in touch with the actual business use cases and helps them better understand the impact that bugs have on the end-users. It also provides the advantage of detecting breaking changes early in the process, keeping high visibility on overall quality throughout the development cycle.

Today, while development teams have the benefit of modern IDE's and object-oriented software development technology, most test teams' automation approaches are largely based on monolithic scripts. Objects created for each the application screens become the basic building-blocks of modern-day test automation and offer the benefit of targeting future changes to very specific elements rather than requiring a rewrite of the entire script, thus making test maintenance effortless and providing the agility required for a Shift Left strategy. However, these tools are far from ubiquitous and QA is often left struggling with old technology.



turnkey

Strategy 3: Accelerate Application Deployment

DevOps reduces the time between concept and deployment and this acceleration elevates the need for test automation and easy test reconfiguration. The question of what to test requires upfront analysis and planning in order to pinpoint areas that represent the largest business risk and value. These tests should also cover parts of the application that have changed, "hot spots" that have been historically problematic and areas where the cost of failure is the highest. Test "Impact Analysis" is even more important with DevOps because there is no time for lengthy test/fix cycles to develop and maintain custom test scripts for every iteration, or to conduct extensive manual exploratory testing.

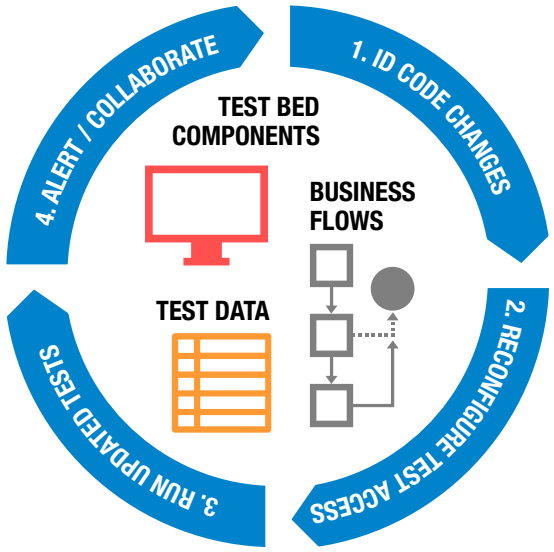
One positive side effect of releasing code in small iterations is identifying when an issue is introduced with more precise granularity. Lengthy iterations of a classic waterfall release cycle make resolving issues more difficult, since so much has changed between the deployment cycles. With small iterations, it becomes easier to pinpoint **when** something has changed. This combined with code change tracking frequently identifies precisely **what** has changed to create the issue.
The following example illustrates how Test Automation can better support DevOps objectives:

1. DevOps teams review planned changes, business goals, and risks. Based on this impact analysis a test plan is developed. Ideally this commences before any code is written for a code iteration. The team generally identifies issues with the workflow or the design during this step.
2. Developers then create scriptless and reusable test components with advanced Test Automation software when first developing the user interface layer. Test data used is just "made up," but sufficient at this point. Issues found within the code base are generally detected here.
3. QA is involved to generate high-quality test data and configure tests into various workflows. As the application becomes more stable, tests are scheduled to run automatically either nightly and/or based on events like code check-in. Issues found within the code base are found here.
4. QA enhances the test suite to include complete business end-to-end workflows with live data. Issues found typically relate to the interactions within the application or application-to-application interaction issues.
5. Operations uses these automated tests for regression testing. Issues generally found here are related to the production environment (servers, databases, etc.).

The biggest challenge to this agility is test maintenance. Keeping tests up-to-date and enhancing the broad test coverage to include new issues is a common challenge when using DevOps. It becomes easier to keep tests in an evergreen state through best practice test automation.

1. After code changes are released, intelligent Test Automation software automatically identifies what has changed so that developers can confirm and accept these changes to the test components.
2. Changes to components are automatically propagated to the test cases and the test sets containing those components.
3. QA then enhances impacted test data and end-to-end workflows.
4. Tests are rerun by all validating the software changes.
5. Troubleshooting of pass/fail results occurs for potential code base issues.

The ease of reconfiguring through Test Automation not only accelerates delivery, but improves coverage over record-and-playback automation or test scripts that represent a single path through the application. This is called path-locking. It then becomes feasible and efficient to test thousands of workflows and data scenarios, instead of just a few.



turnkey

Strategy 4: Add Scriptless Test Automation to Your QA Stack

In a DevOps world, where agility is key, emphasis is typically placed on speed and efficiency. To achieve this objective, automation is a key determining factor. For many, deployment tools like Puppet Labs®, Chef®, Jira, Jenkins, and others ARE DevOps. While deployment automation is very important, the ability to quickly test and certify the functional quality of any new or updated version is equally, if not more important, than the speed at which it can be deployed. Since Quality Assurance is often the most time-consuming and costly part of any software development project, even small scale reductions in QA have significant impact.

Today, the vast majority of the software QA automation tools available were developed in a different era and for a much different use case: waterfall testing processes. **Gone is a world where developers bear the full responsibility and burden of testing, and gone is a time when QA teams had months for a testing phase in the project.** DevOps requires QA solutions that are extremely agile, allowing for tests to be easily and rapidly maintained and adapted to the needs of a new iteration.

Requirements for DevOps Compatible Test Automation

A modern Test Automation application closely fitted to the DevOps world must meet 3 key requirements:

1. **Usability** – Simplifying test development for everyone on the DevOps team
2. **Reusability** – Component based & keyword driven, so assets can be used for multiple applications and end-to-end tests
3. **Data-Driven** – Supporting multiple scenarios from a single test

Usability

"Scriptless Test Automation" not only extends the ability to create and modify tests beyond the development group, but more importantly, it decreases the time to develop driving forces in making automated testing work in a DevOps model. Without the speed and efficiencies gained by leveraging this type of solution, it becomes next to impossible to support testing at the velocity a DevOps model requires.

Another way to enhance speed and agility is through "application aware" testing. Each application has its particularities. This entails that tests need to be designed differently for each DevOps project. Application aware testing helps to automatically create tests and update keyword-driven test components based on application specific parameters and best practices. With application aware testing, components are assigned keywords that indicate the operation associated with a given object on a screen. This capability not only significantly reduces the time it takes to create tests, but also detects what has changed, allowing tests to be updated quickly and efficiently for each new iteration, ensuring minimal delays in the DevOps process.

Reusability

Best practices that apply to mitigating risk and minimizing maintenance in any software development project naturally apply to the test cases as well. It is widely accepted that development teams can achieve flexibility and rapid development cycles through a object-oriented approach to the software development lifecycle. Having a object-based Test Automation platform provides the same benefits for testing. Modularity in tests allows for maximum reuse and minimal maintenance, not only in the ability to test different workflows with minimal reconfiguration, but also to quickly configure tests to handle modifications to existing workflows when changes occur.

"Keyword driven" is a process of separating test objects from test actions. Each object in the application under tests has a set of actions that can be done against it as well as a default action. Using this approach in test creation is greatly simplified and standardized as the complexity is encapsulated in the automation software.

Data-Driven

Test Data Management **refers to** leveraging a Data-Driven approach - separating tests from the data used to drive test flows. This method provides the agility to run multiple data sets against the same test and is critical for increasing test coverage and agility. This separation is sometimes criticized because of the additional time it can take to configure upfront versus using a scripted, hard-coded approach. Scriptless Test Automation in TurnKey's cFactory product is designed from the ground-up to reduce the up-front configuration time, gaining all the advantages of using robust data sets when testing. Using cFactory, business users who understand workflows and data can easily and rapidly design complex test scenarios without requiring any scripting or coding knowledge whatsoever, thereby lessening reliance on expensive development resources.

Strategy 5: Identifying A Business Case for Scriptless Test Automation

Test Automation is not only a cornerstone of any DevOps quality initiative; it is required to establish Continuous Delivery without compromising quality. In addition to better collaboration, Shift Left, and accelerating application delivery, one should also identify a strong business case for Test Automation to reinforce its critical role in DevOps transformation. This includes a focus on hard dollar cost justification, additional business benefits, and retooling processes. Scriptless Test Automation has proven to yield highly compelling ROI numbers while helping to drive improvements in vital QA Key Performance Indicators.

Cost Justification

As previously discussed, the cost of a software defect that is deployed is far greater than the time to find and fix it early, so an obvious place to begin is lower QA labor expenses – whether onshore or offshore. Scriptless Test Automation provides order of magnitude improvements in the utilization of QA labor, leading to compelling breakeven analysis.

Additional larger costs to be avoided include:

Current Cost = missed market opportunities + QA delays + customer support + brand reputation damage

Note that the largest cost of poor quality is not in people's time, but in the impact that poor software quality has on your business and brand. No Test Automation strategy can eliminate all these expenditures, but if the current costs are high enough even a single digit percentage gain in test efficiency and effectiveness will yield substantial savings.

Business Benefits – Key Performance Indicators

Apart from hard dollar savings, there other compelling business benefits to Scriptless Test Automation, and TurnKey Solutions suggests the utilization of at least 4 key QA KPI's as a baseline to measure ongoing QA improvement; including:

1. Decrease Mean Time to Repair (MTTR)
2. Decrease Time to Test (TTT)
3. Reduce Time to Market (TTM)
4. Reduce the number of issues "escaped" from the DevOps process

Retooling QA Processes

With a move to DevOps, the role of QA shifts from primarily detecting defects, to supporting a Continuous Delivery environment that prevents defects in the first place. By leveraging a Scriptless Test Automation platform like TurnKey's cFactory software, QA teams, Business Analysts, and product owners work closely together and establish tighter visibility and control over the test configurations and test data. Development no longer waits for lengthy feedback loops and can run end-to-end tests on the full stack to gain insight on exactly how their changes have impacted quality. Leveraging this strategy, Test Automation becomes a foundational element that helps to bring QA and Development together to enable a Continuous Delivery discipline within the Application Development Lifecycle.

About TurnKey Solutions

TurnKey is the leading provider of scriptless test automation solutions. With over 20 years of experience in bringing test automation tools and technologies to market, the company offers a portfolio of products designed help companies accelerate delivery, lower costs, and meet the demands of a more dynamic testing environment.

More than just improving efficiency across the software development lifecycle, TurnKey Solutions gives businesses the intelligence they need to respond to customer needs and maximize the quality of the applications for the end user.

Visit www.turnkeysolutions.com for more information or contact sales@turnkeysolutions.com to schedule a live product demonstration today.

1 http://techbeacon.com/10-companies-killing-it-devops

2 Vance, Stephen Quality Code: Software Testing Principles and Practices p5: Addison-Wesley, 2013

turnkey