Special Report

The Role Of DevOps in Increased Business Agility and ROI

HEXAWARE

Whether you are a seasoned DevOps veteran or you have just started your DevOps journey, at some point you will need to decide whether you are getting the most value from your investment. Turning a critical eye toward your tools and processes is never easy, but today's hyper-fast delivery cycles demand smart business decisions.

This DevOps Special Report will help you pick the best tools, processes, and technology to maximize business agility and measure the return on your DevOps investment.

In this DevOps Special Report

HEXAWARE

Application Release Automation: Why the QA Pro Should Care

By Tracy Ragan

Testing is right smack in the middle of the DevOps conversation. As most QA professionals have noticed lately, continuous testing and test automation are critical pieces of the continuous delivery pipeline. What most QA pros also understand is that the speed of testing depends on a consistent software release process that can provide the critical information needed when reporting issues.

QA pros will benefit from a new set of DevOps tooling called application release automation (ARA). These tools focus on the modeling and deployment of software releases and their associated configurations, supporting continuous release deployment. ARA allows the QA pro to clearly see the changes moving into and out of the testing environments, providing visibility into what was deployed and updated.

Common to most continuous delivery pipelines is the process of installing updates to the testing environment, or at least to a subset of servers defined for executing the test process prior to opening up those servers for end-users. While test automation can speed up the actual execution of the test, the deploy and install process can be riddled with hidden updates, inconsistencies, and no solid reporting to quickly determine what version of an artifact may be the cause of the issue.

In order to drive that continuous delivery pipeline, both the software compile and the software deployment must be transparent and understood by all teams.

When you are trying to go fast, you need some guardrails that will allow you to quickly determine a deployment issue, even before you bother running those automated test. After all, if the deployment is bad, testing will fail.

According to some, continuous build means that you should be able to fix a build within ten minutes in order to get the CI/CD workflow back up quickly. Continuous deployment is the same. In order to drive that continuous delivery pipeline, both the software compile and the software deployment must be transparent and understood by all teams.

ARA solutions are the drivers of continuous deployment. For the QA pro, it means that they have the ability to control their testing environments, understand the changes coming their way, see what updates have been applied, and provide some level of continuous feedback. A solid ARA solution will connect your change request, issue ticket, or user story to a particular release version. QA pros can quickly understand what updates are on their way and determine the level of risk associated to the release.

Ultimately, the QA pro must be the one to push that change forward to production. With an ARA tool, the production release can be simplified by being, in effect, a repeat of the test release. Once you have this level of repeatability and visibility, your job as the mediator between dev and prod becomes much easier, and your continuous testing can really fly.

HEXAWARE

Why DevOps still Needs Release Management

By Adam Auerbach

Release management entails planning, scheduling, and controlling a software build through different stages and environments. And it comes in many different flavors.

In many cases release managers are the gatekeepers of the change management process, making sure production deployments are well orchestrated and follow all the necessary steps to ensure the proper visibility and approvals are obtained throughout the process. They might also run the various change advisory board meetings and actually run the production implementation events.

In these examples, release managers typically have a project management background. They are experts at process and communication, which makes them a good fit for this flavor of release management.

In some cases, release managers are more like engineers. They have the responsibility and own the deployment process itself by executing deployments on behalf of development teams and playing a role in automating the deployment process.

As companies look to adopt a DevOps philosophy, the role of release management has to shift as well.

In DevOps, we allow teams to have control over production deployments. The team is not just focused on creating working code, but also on the infrastructure, network, and other implementation items necessary to get their code into production. By having this accountability, teams will create code with higher quality, making our production systems more reliable and maintainable. But as my Uncle Ben says, "With great power comes great responsibility."

So if teams are practicing DevOps, does our existing release management process become obsolete? The answer is no, but it does need to change.

Having change orders or some record of change is still needed in DevOps because it's not just needed in reg-

ulated environments to show traceability from code to deployment; it helps shed light on release volumes and time-to-market metrics, which we know are core to measuring our DevOps maturity. However, what will change is the information captured in a change order and how these change orders are created. You will no longer need to track implementation or back-out plans as part of change orders. You just need to be able to track the application, its components, and its promotion schedule.

As with many things, the key to maintaining these change orders is automation. Your continuous integration pipeline has to have the ability to communicate with your change order system so self-documentation can occur.

Creating communication between your delivery pipeline and your change management system gives you the ability to also prevent releases. I know that might sound strange. You might get to the place where you can deploy without restriction, but that won't be early on in your DevOps journey. There will be holidays or some other instances when you want to limit the number or type of production releases. Having an automated ability to communicate with your pipelines will allow you to have a way to throttle deployments based on business need. This is done today via freeze windows, change advisory boards, and other go/no-go type meetings, but in a DevOps environment, you need a real-time way of doing this directly with the pipelines that eliminates human interaction.

One of the biggest opportunities with release management being enabled via a tool is that you can integrate your audit and security requirements into your process. For example, many companies have controls around separation of duties, test traceability, and business approval. With a tool, we can automatically ensure that the person checking in the code is different from those who have access to the pipeline or to approve the release. For testing, we can ensure that we have gates for testing coverage and security testing findings. No code would be eligible for promotion if it does not meet those thresholds.

Rather than doing audits of releases after the fact, with a tool, you can integrate these controls as part of the pipeline. Thus, they become first-class citizens and you ensure that risk is actually prevented versus discovered reactively. With DevOps and automated release management, you can build a system that is better managed than how the current audit process works.

Because this is a journey, most likely your newer web and mobile applications will be quicker to get to continuous delivery (CD) than applications built on more

Having an automated ability to communicate with your pipelines will allow you to have a way to throttle deployments based on business need.

traditional platforms. In reality, you will be operating in a mixed model, with both manual and automated deployments, for a while, and you need to have an approach that will work for everyone as well as provide insight into the DevOps maturity of your applications. Continuing to use a production change request (PCR) tool will give your leadership transparency on the success of CD and where the opportunities still exist to improve the delivery process.

The role of release management in DevOps needs to focus much more on how to enable automation and integration.

For those who still have process-focused release management, you will need to provide a path forward to make a transition to this model. For those who have some technical aptitude, you will want to provide opportunities to learn and experiment so that they can have the option to transition to an engineering role.

You will need to socialize internally and with your customers about the new direction and vision for the team. That will energize those willing to learn and transition, while others who aren't interested will have opportunities for other roles in your organization. In the end state you might not need as many process-focused team members, but you will need some, as the process will always require some level of oversight.

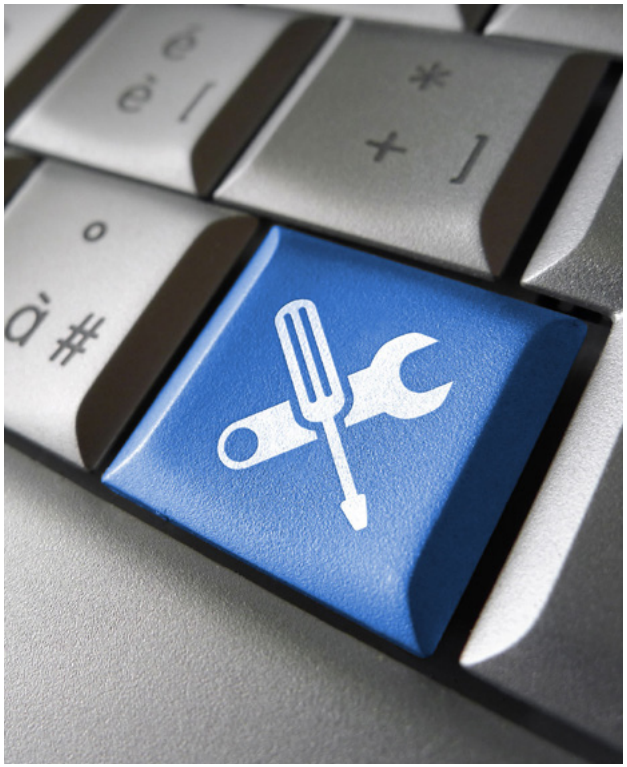
The role of release management in DevOps needs to focus much more on how to enable automation and integration. For the release management function to stay relevant during and after your DevOps transformation, release managers need to move away from viewing release management as a project management-type role and into a more technical role that builds or configures an API-driven tool that integrates into your pipelines and PCR tool and itself embraces a DevOps approach.

Release management is still critical in a DevOps environment. Its function also has to drive the shift from a service-based organization to an engineering group that enables frictionless flow to production via a kind of virtual air traffic controller for production code. This way, you create a "you build it, you own it" model with a better level of security and oversight.

HEXAWARE

A Guide to Selecting the Right DevOps Tools for Business Agility and Optimized ROI

By Lakshminarayanan RS—Solution Architect, Hexaware Technologies



DevOps tools are not only a means to create products, they also play a crucial role in automation, which directly ties to consistency, repeatability, and faster time to market. Thus, it is important to understand the role of these tools. There is no-one-size-fits-all tool to address the challenges you encounter during your DevOps journey. You need a structured approach to selecting the right tools and methods to optimize your DevOps ROI.

This white paper explains what DevOps is and why the right tools are crucial. We also highlight Gartner's three categories of DevOps tools and explain how to select the right tool from the plethora of tools available. To help you make an informed decision, we list a few parameters to consider while selecting a tool.

In the later pages of this white paper, we show you how to check your organization's readiness to adopt DevOps and to prepare a roadmap for adopting this methodology in the best possible manner.

DevOps at a Glance

Technology disruption is a threat to every industry. The implication is that businesses have to think far ahead and push their IT teams, which in turn need to be creative to build products with a shorter time to market. The businesses need to sustain this momentum, and IT has to follow suit every time. The processes lay out the framework, and teams build in increments to release a minimum viable product in a shorter time frame. Feedback mechanisms are in place to constantly improve the product quality over time.

DevOps is a methodology that encompasses people, processes, and tools to achieve business agility, thereby shipping software products faster to market. It is a technique that automates the process of software delivery and changes in infrastructure while bringing a sound coordination between the tasks carried out by software developers and the rest of the IT team. Selecting the right tools for DevOps is crucial as it directly influences not just the product development process but also the time to market of the product.

The tools also aid in team collaboration and in creating process templates that can cover the application lifecycle management (ALM) aspects of the product.

Many organizations have already adopted DevOps for their business environment while others are keen to understand how they can transform their business by using this technique or are waiting to learn from early adopters before making up their minds to use DevOps.

Below are a few areas where DevOps scores over traditional IT Ops:

- DevOps-oriented teams spend 21 percent less time each week rolling out changes
- DevOps-oriented teams spend 33 percent more time improving infrastructure to avoid failures
- DevOps-oriented teams require nearly 60 percent less time per week to handle support cases

There is no standardized guide to selecting the right tools for implementing DevOps techniques. To be successful, an organization needs to do a reality check of its business before adopting DevOps. Organizations first need to thoroughly review the tools they already use and assess how they are using them. To do this, finding answers to the following questions can help:

- What takes more time: build, provision, or deployment?
- Where is the quality hit: security, performance, or scalability?
- Does the choice of tools make sense for continuous integration/continuous delivery (CI/CD)?
- Is there sound collaboration between development and other IT operations?
- How are the operations monitored and feedback gathered?

Gartner's Three Categories of DevOps Tools

According to Gartner, DevOps tools can be divided into three major categories: DevOps ready, DevOps enabled, and DevOps capable. A brief explanation of each category is given below.

1. DevOps-ready tools are out-of-the-box solutions designed to support at least one of the workflow steps in DevOps. These tools offer application release automation and continuous configuration automation. *Examples: BMC, VMWare, Chef, Puppet, Ansible*

2. DevOps-enabled tools are designed to work in a pipeline environment and enable activities for development, testing, quality assurance, and production for the application and infrastructure. The tools focus mainly on integrity and fidelity of the application and infrastructure. These may not be new technologies, but they have direct extensibility for DevOps projects. Tools included in this category perform functions like continuous integration, continuous quality, code review and static analysis, test automation, and environment/pipeline management. *Examples: Jenkins, Atlassian, Microsoft, Travis CI, Delphix, HP, IBM, CheckStyle, Coverity, SonarQube, Veracode, Cucumber, FitNesse, CloudBees, Electric Cloud, ElasticBox*

To be successful, an organization needs to do a reality check of its business before adopting DevOps.

3. DevOps-capable tools have typically been around for years, but they usually fall into another IT operations management category. However, these standalone tools can work in a DevOps pipeline when configured correctly. Monitoring, security testing, and lab management tools fall into this category. Some newer monitoring tools include characteristics and traits of DevOps tools and are therefore more easily adopted specifically for DevOps projects. *Examples: AppDynamics, New Relic, Datadog, Elasticsearch/Logstash/Kibana, Ganglia, Splunk, Veracode, PortSwigger, N-Stalker, OpenVAS, Microsoft, IBM, CollabNet, Amazon, Dell*

Selection of the Right DevOps Tool Made Easy

There is no rule of thumb for selecting a DevOps tool. The selection depends mainly on the nature and scope of your product as well as your prime requirements. The tools can be from either or all of the Gartner's categories mentioned above.

The following tips can help you select the right tool:

- If there is sufficient exposure to the cloud, then cloud-based DevOps tools like Azure VSTS, AWS, or Google App Engine can be used. Some organizations have begun to adapt open source PaaS vendors in this category like Cloudify, OpenStack, etc.
- If on-premise hardware and servers are available for development and QA, then deployment can be automated by maintaining configuration in tools like Chef, Puppet, or Ansible. Jenkins can act as a CI tool.
- ALM tools like Team Foundation Server, Atlassian JIRA, and Rally are required to address collaboration between development, business, and operations teams. Open source vendors that participate in this space are Tuleap and OrangeScrum.
- Jenkins is the prominent CI tool and has a collection of plug-ins that integrate with a wide variety of tools from Java, Microsoft, and open source platforms. Other CI picks are Travis CI and Go Strider.
- To perform security and performance tests, open-source tools like Metasploit Framework, Brakeman, Cuckoo Sanndbox, JMeter, TheGrinder, Gatling, and Tsung can be explored.
- If there is a need to maintain configurations specific to various environments (development, QA, UAT, and production), and if the machines need to be engaged and decommissioned as per requirements, there are open source tools available like Chef, Puppet, and Ansible. If using Microsoft Azure, then the release management and lab management aspects of VSTS would fit.

There is no rule of thumb for selecting a DevOps tool. The selection depends mainly on the nature and scope of your product as well as your prime requirements.

- Feedback mechanisms can be configured using Microsoft's Application Insights from VSTS or open-source tools like Nagios and New Relic.
- Google Cloud Platform products like App Engine, Compute Engine, and Container Engine can be an option if Google is the cloud provider.
- Reporting and dashboard in DevOps is a complex undertaking. These ALM vendors—VSTS, JIRA, Rally, HP ALM, Clarity and ALMComplete—have built their own tools to derive meaningful project lifecycle management like overall status, team velocity, sprint burndown, enhanced release burndown, sprint interference, and remedial focus.
- Another aspect of monitoring is application health, which is a crucial element of overall business success. There are various tools like Nagios, New Relic, and DataDog can drill down to hardware and server parameters like internal thread count, Disk I/O, Inx Read/Write Operations, etc., that will serve as a dashboard for the administrators.
- Each of these tools produces metrics in its own way. In order to understand from an ALM perspective, the customer needs to switch between the data views and manually needs to figure out both project and application health indices. Hence, it is imperative to have a unified view of a reporting dashboard across ALM and monitoring tools. Hexaware has identified ELK (Elastic, LogStash and Kibana) as a unified dashboard monitoring tool that aggregates across multiple data sources produced by different tools. Logstash is a tool for managing events and logs and can be used to collect logs, parse them, and store them for later use (e.g., for searching). Elasticsearch is a search server based on Lucene. It provides a distributed, multitenant-capable full-text search engine with a RESTful web interface and schema-free JSON

documents. Kibana is a nifty tool to visualize logs and timestamped data.

- The world is moving toward Microservices, so it is important to get the thinking right in terms of choosing containers like Docker and envisioning how other configuration management tools like Chef, Puppet, Salt Stack, and Ansible fit into the DevOps chain. With the given set of tools integrated and running in the deployment pipeline, it is important to have a disaster recovery environment setup planned for the same.

DevOps Metrics

There cannot be success without measurement. It is important to gather metrics like deployment frequency, lead time, percent of failed deployments, mean time to recovery, ticket volume, and percent change in user base from the tools and measure them periodically. The deployment frequency is easy to measure as this depends on the release plan and business criticality. Lead time is the time taken for the new code to move from development to production. Optimizing the lead time will result in better time to market. It is also important to keep a check on the percentage of failed deployments. MTTR is more about the product's robustness and resilience. High percentage of ticket volume indicates that there are more issues incurred. Change in user base is the percent of new users to the product. Most of these metrics can be obtained using DevOps tools.

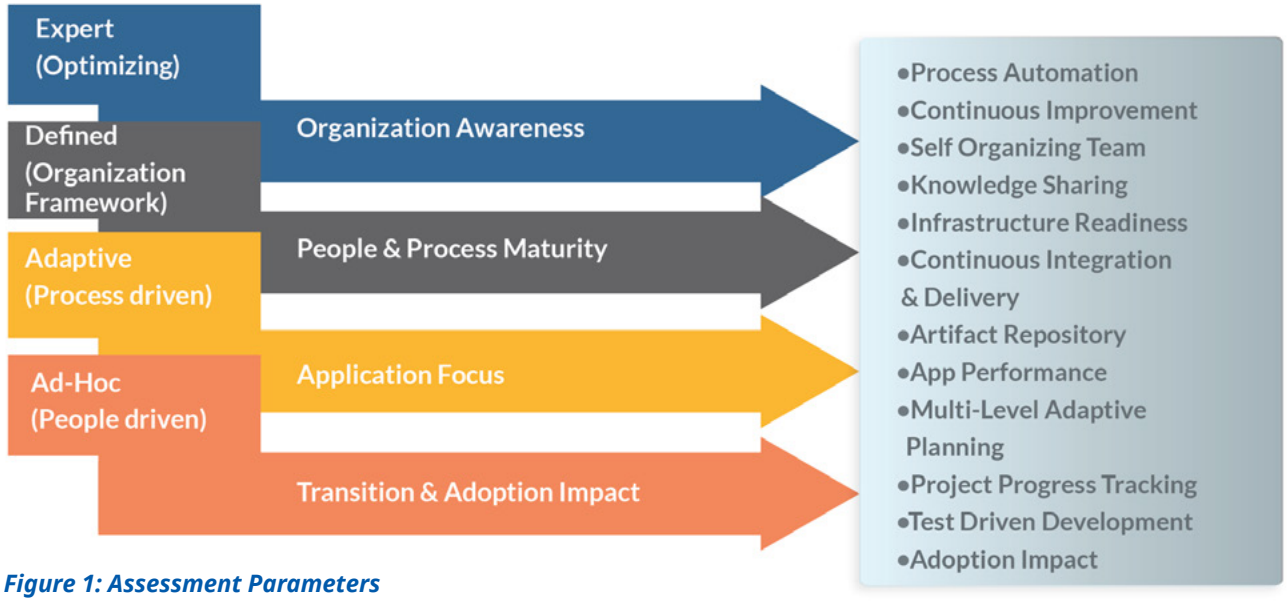


Figure 1: Assessment Parameters

Formulate a DevOps Automation Strategy

It's important to assess an organization's readiness to adopt DevOps at a portfolio, program, or project level, then create a roadmap and next steps.
The assessment process is elaborated in four steps:

Step 1: Determine the DevOps maturity
The objective here is to determine the team culture and the level of adaption the environment in question may have with respect to release management automation and reporting. Figure 1 shows the parameters used to assess maturity.

Here is a high-level overview of the assessment parameters in figure 1:
Organization Awareness: Covers the organization's cultural characteristics and its ability to adopt process automation successfully.

People & Process Maturity: Covers exposure and acceptance levels of involved stakeholders (managers and implementation team members) for adopting DevOps automation tools. It also assesses maturity of "process automation tools" to meet technical expectations (nonfunctional requirements).

Application Focus: Evaluates how process automation can help in the development of software.

Transition & Adoption Impact: Tries to gather data points around the transition required for automation and plausible areas of impact.

Step 2: Determine technology stack and map tools
A prioritized list of projects from the portfolio inventory is used to determine the technology stack to be used to formulate a strategy for DevOps automation. Based on this, the tools for SDLC/PLC are identified in line with the organization's IT strategy and customized per the requirements. Some DevOps lifecycle areas to consider when aligning tools and identifying respective challenges for each product or program include: ALM, collaboration, deployment, code review, and functional testing.

Step 3: Record tool usage pattern
Steps 3 and 4 are executed when a minimal state of automation exists or during steady state for audit/health checks. In addition to studying tool metrics, it is important to understand how these tools are being used. For example, if there is a source control tool being used on a monthly basis, then it is being underutilized. Or, if there are no automated builds, then developers have to spend more time manually building the product. The metrics note the tool usage frequency along with the effectiveness score. Depending on the frequency and the score, a score card index is calculated using an appropriate method. The metrics of tools data will result in many possibilities. The tools used in one product may not be a reflection of tools used in other products

Step 4: Analyze tool effectiveness with respect to fulfilling DevOps objectives
The objective here is to determine whether the tools and processes being used are actually meeting the desired maturity level. During this step you will assess and track the metrics that were identified and agreed upon in step 1 and the goals are set against each metric respectively. You also have an opportunity to identify current bottlenecks in enabling DevOps as well as areas of improvement. This allows you to set a new target maturity benchmark when the outcome determines that a particular targeted maturity level has been sustained for an agreed timeframe.

Figures 2 and 3 show sample outcomes of the assessment.

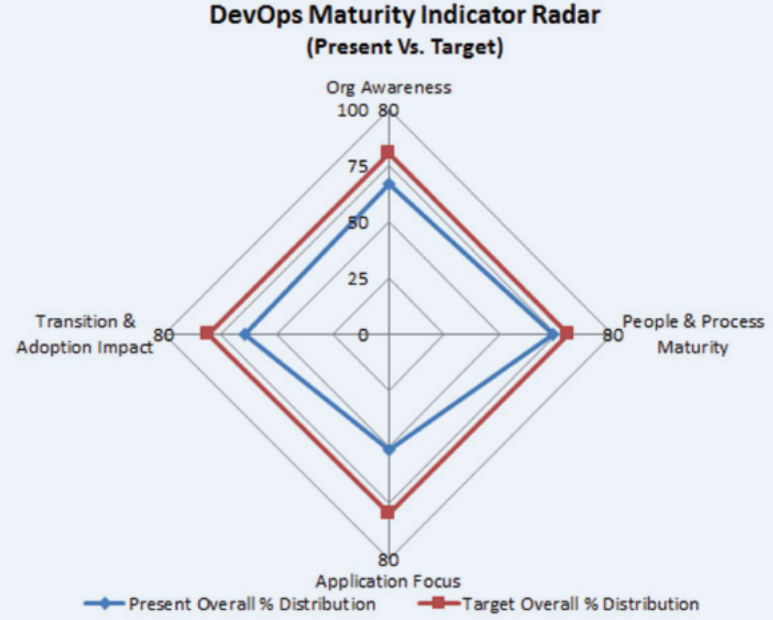


Figure 2: DevOps Maturity Indicator Radar

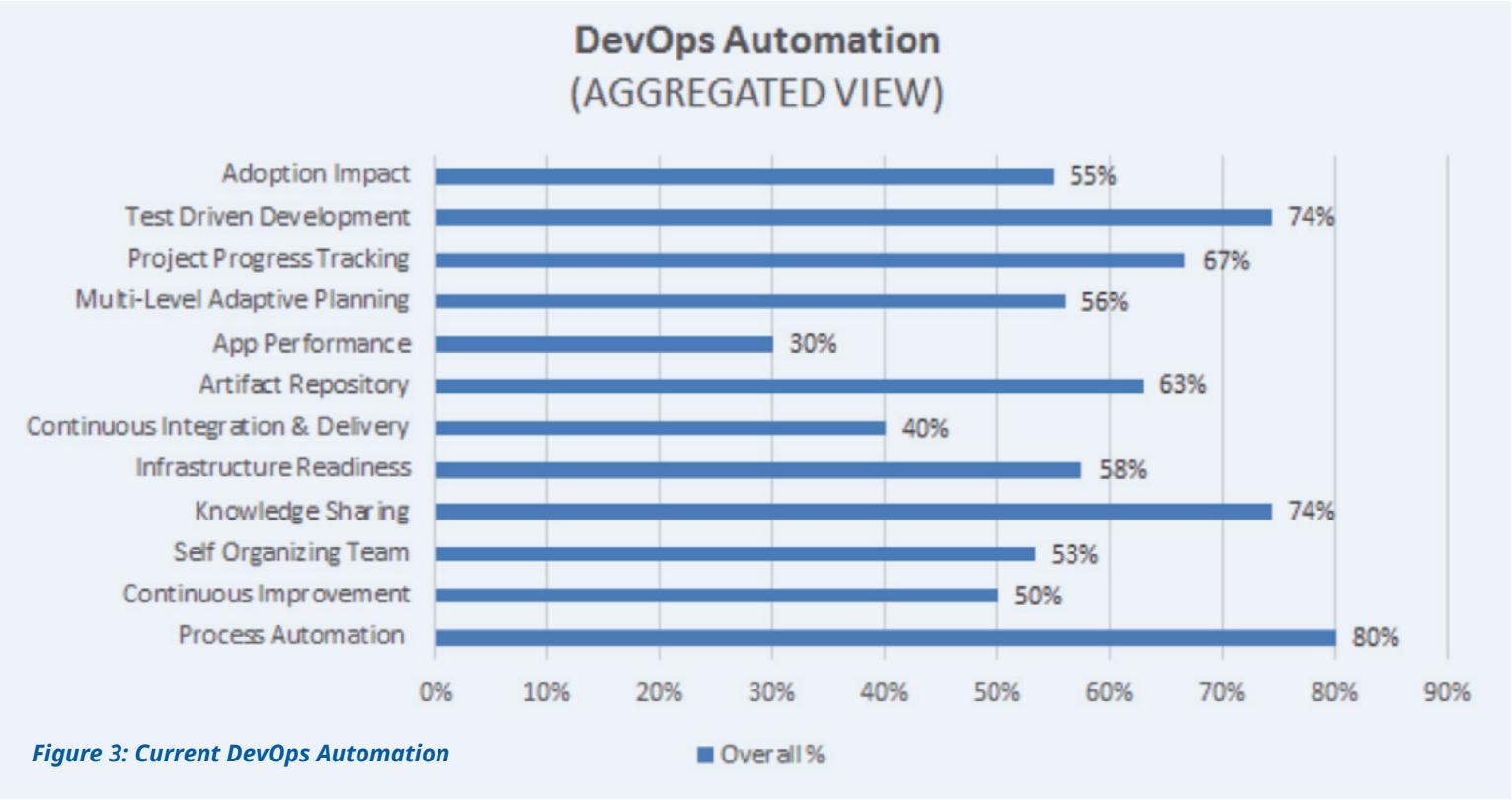


Figure 3: Current DevOps Automation

Take Stock of Your DevOps Tools

DevOps is shaping up fast and has reached a measurable stage where tangible and intangible benefits can easily be seen. The key lies in understanding the premise in which a business is being operated and how the right tools can optimize the deployment. It is always beneficial to align new tool choices with the existing toolset so as to sustain your future scale and growth without making unreasonably huge investments. In the end, the most important objective is to achieve desired time to market with efficiency and precision.

Click Here for References

It is always beneficial to align new tool choices with the existing toolset so as to sustain your future scale and growth without making unreasonably huge investments.

HEXAWARE

Test Your Data Quality to Increase the Return on Your QA Investment

By Shauna Ayers

With the high volume of data coming into your organization, it's important that the information be complete, correct, and timely. The consequences of mismanaged data can be lost revenue opportunities, insufficiently informed decisions, and decreased customer satisfaction.

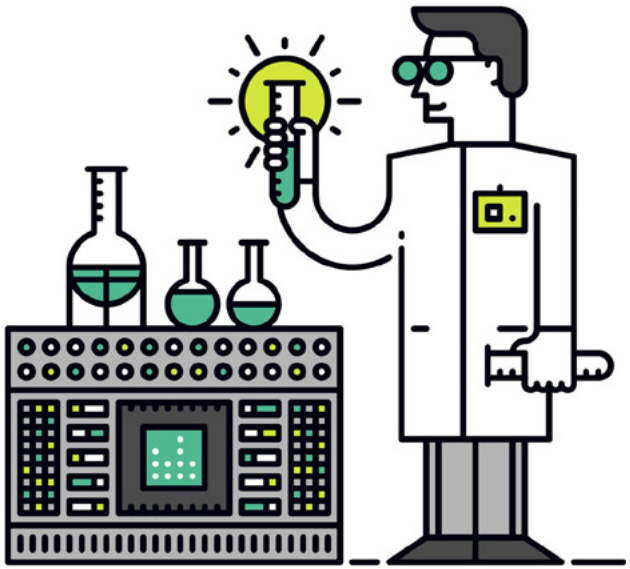
But considering the velocity at which this data is moving, how do you measure its current quality? You must be able to test it wherever it sits still enough to be viewable, without altering it.
To show data quality (DQ) risks to the decision-makers in your organization, you must be able to retrieve data obtained through these tests and channel it into standard reporting mechanisms. And to quantify the return on investment of your data quality program, you must be able to measure data quality, and any changes in that quality, over time.

To do all of this, you need repeatable, reproducible measures stored in a format easily consumed for reporting and analysis. Data quality testing can provide these measures. Centralizing the collection and storage of DQ test execution results facilitates automated production monitoring of the health of your data streams, including dashboarding and proactive alert notifications. For instance, automated DQ checks can trigger alerts about issues that require immediate attention, such as runaway conditions, tolerance thresholds being exceeded, or security issues only visible through the data.

Using DQ test execution data to fuel communication and trends analysis in the data flow can lead to measurable process improvements, which can increase the return on investment in your data quality program.

HEXAWARE

Data quality testing does not reside solely within the realm of software development. You must also test and monitor the quality of your operational data in order to protect your organization from costly errors.



Data quality testing does not reside solely within the realm of software development. You must also test and monitor the quality of your operational data in order to protect your organization from costly errors introduced by factors occurring outside the development cycle. You can do this using a variety of tools that allow reading and sampling of data without altering the data itself.

Regardless of the tool you use, funnel its test execution results into a centralized data quality database. As systems and platforms change, the tools used to view and monitor their data may change. Unifying their output in a single database allows your data quality program to be robust in the face of these changes.

Many tests can be automated and run on a schedule, including source-to-target count checks that monitor ETL processes, domain integrity checks that monitor reference data management, and environment checks that provide snapshots of process lag and operational bottlenecks within the flow of data. These test results can reduce operational costs and avoid progressive degradation of performance:

- Use time-sensitive results data to provide proactive notification alerts when an issue is detected
- Classify results by subject or scope to power dashboards showing the current health of the data system, or identify and quantify the recurrence of chronic issues
- Analyze results over time to identify patterns that point to resource and capacity issues, hardware decay or load balancing problems

To keep your organization running at optimum speed, protect the health of your data by leveraging data quality tests to their fullest.

Building a Business Case for Automation in Your Software Lifecycle

By Michael Sowers

We continue to produce new and enhanced software capabilities at lightning speed. This is a result of many decades of learning and evolving our software engineering discipline, adopting more responsive methodologies and tools, employing better programming languages, and embracing the principles and culture of agile and DevOps, among other things.

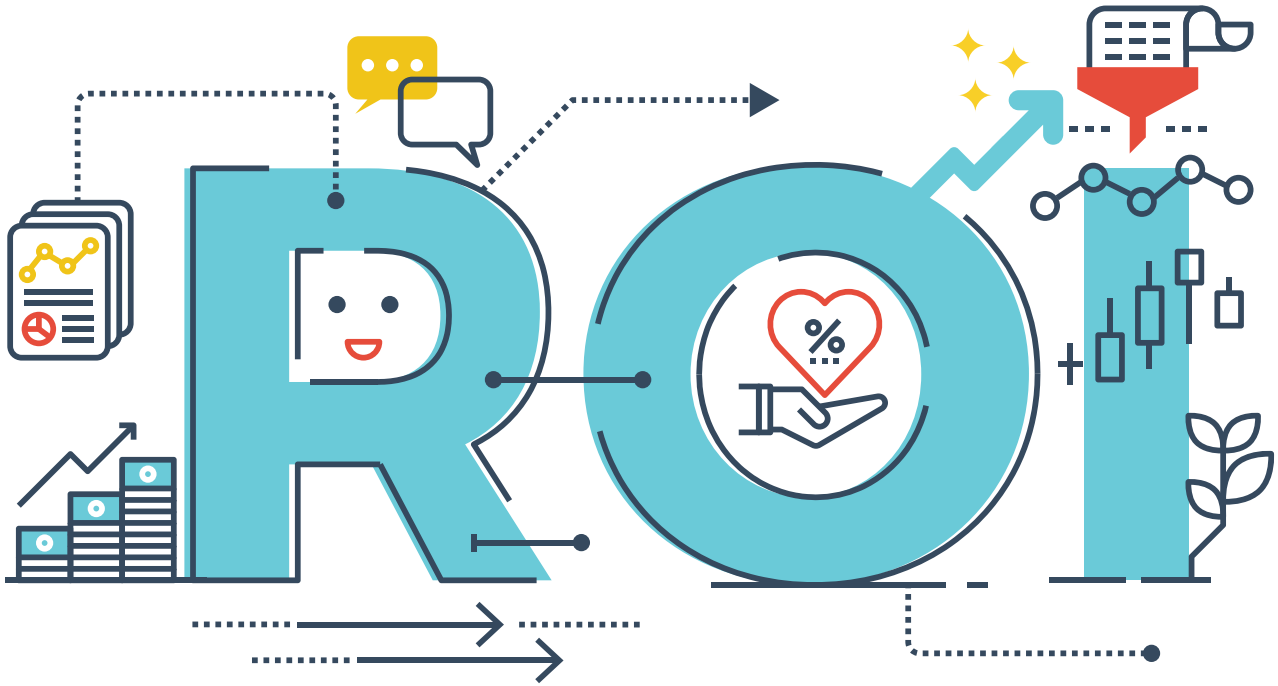
In the midst of this necessary and continual transformation of our profession, it surprises me that I continue to get the question "How do we sell automation?" Usually it's focused on "How do we convince our organization that automating testing is beneficial?" but I have also heard this question regarding the automation of other aspects of the software engineering lifecycle.

Let me preface my comments by agreeing that, in an ideal world, we would deliver perfect code, continuously, that needed little or no testing. While a few organizations are getting there, the majority of us are still faced with the reality that we must have some testing in place to mitigate business risks. Returning to the question of selling automation, my first reaction to that question is, "You're kidding me, right? We're willing to invest in all kinds of automation in other aspects of the lifecycle, but we need a detailed business case for test automation?"

OK, enough venting! My overarching belief is that the organization that ignores the implementation of a well-integrated set of automation capabilities across the entire lifecycle compromises its ability to remain competitive.

Here are a few things to consider as you assess your automation investment. I've focused this on test automation, but many could be restated to apply to automation elsewhere within the software lifecycle.

Qualitative questions: What are the existing pain points, and how would automation mitigate them? How would automation free those in testing roles to do other, deeper testing? What's the benefit of the agile team having immediate feedback when a change is implemented? Is there a benefit of being able to run



tests 24x7? Will automation contribute to internal or external compliance requirements? How important is test repeatability and consistency?

Quantitative questions: What are the current ranges of time and cost required to:

- Plan, design, implement, execute, and maintain test cases?
- Plan, provision, and maintain the test environment(s)?
- Determine, create, extract, mask, manage, and maintain the test data?
- Manage the existing manual test cases?

You should start with a small business case for one specific type of testing that significantly addresses a critical pain point, and see what kind of benefits automation brings.

Having this data will tell us how many more test cycles and test cycle variants could be executed if automated. Benefits include faster test execution, less time provisioning environments and test data, ease of managing test cases, improved test status reporting, quicker feedback to the team when issues are found, reduced stress on the team, greater consistency and repeatability in testing, the ability to run more tests (with different data sets), and increased test coverage.

Don't forget that the ROI for automating different types of testing will vary, such as between regression tests versus user story tests. You should start with a small business case for one specific type of testing that significantly addresses a critical pain point, and see what kind of benefits automation brings.

HEXAWARE

Insight from around the Industry

On Why You Should Invest in DevOp

"A few organizations have made DevOps part of their day-to-day and it has paid off quickly. Companies that have not started yet are behind and should definitely be putting time, effort, and money into DevOps. Our clients who invest in doing DevOps correctly early on see huge dividends in flexibility, speed, and scalability."

—Martin Chikilian

On the Mainstreaming of DevOps

"[DevOps] was like the new thing that everyone wanted, but they didn't know what it was or why they wanted it, they just wanted it. I think it's key to understand why, what it is, do I need it, and how does it apply."

—Jennifer Bonine

On Implementing Agile at Large Companies

"At the end of the day, this work is going to create efficiency, so we should have even more time. It's a dual-headed monster, but at the end of the day, you need to do it or our skills are going to be old, as a company we're not going to be innovating."

—Adam Auerbach

On Embracing Change

"One pain indicator for agile is poor quality, poor time to market, etc. Everyone has to agree that our pain to get over that hump of adoption."

—Bob Galen

On How CI/CDand Agile Impact Business

"While the [CI/CD] tools five years ago were nonexistent, now everybody is rushing out and they really are getting more solid. The more interesting question though is really about the business. It is an amazing change for the business to have a development organization that can do this."

—Jeff Morgan

On Using Docker to Enhance Testing

"Docker is not a one-size-fits-all solution. It has benefits, and if somebody says, "I must implement the Docker," be very wary. But, there are a lot of places where it can be bent to be useful. So don't also just dismiss it out of hand, too. But not, "If you have a hammer, every problem looks like a nail." Not every problem is a nail."

—Glenn Buckholz

Additional Resources

MORE INFORMATION FOR SOFTWARE PROFESSIONALS

VIDEOS

- Hexaware's Big Data Testing Solution—JUMBO
- Hexaware's Integrated Design to Execution Automation Solution—iD2E
- Hexaware's Automation First Approach Strategy
- Hexaware's Multi-Channel Testing Solution
- Hexaware's Continuous Assurance Platform
- Digital Usability Assurance Lab—Hexaware

WHITE PAPERS

- Hexaware-Digital Assurance & Testing services
- Exploratory Testing—A Heuristic Testing Approach

