PARASOFT

Issue 1

DevOps Requires "Continuous Quality"

Why Agile Development Teams Must Reinvent the Software Testing Process

In today's economy, businesses create a competitive edge through software and every company is essentially a software company. Now that rapid delivery of differentiable software has become a business imperative, software development teams are scrambling to keep up. In response to increased demand, they are seeking new ways to accelerate their release cycles—driving the adoption of agile or lean development practices such as DevOps. Yet, based on the number of software failures now making headlines on a daily basis, it's evident that speeding up the SDLC opens the door to severe repercussions.

Organizations are remiss to assume that yesterday's practices can meet today's process demands. There needs to be a cultural shift from testing an application to understanding the risks associated with a release candidate. Such a shift requires moving beyond the traditional "bottom-up" approach to testing, which focuses on adding incremental tests for new functionality. While this will always be required, it's equally important to adopt a top-down approach to mitigating business risks. This means that organizations must defend the user experience with the most likely use cases in the context of non-functional requirements—continuously.

In order for more advanced automation to occur, we need to move beyond the test pass/fail percentage into a much more granular understanding of the impact of failure: a nuance that gets lost in the traditional regression test suite. Continuous Testing is key for bridging this gap. Continuous Testing brings real-time assessments, objective go/no-go quality gates, and continuous measurements to refine the development process so that business expectations are continuously met. Ultimately, Continuous Testing resets the question from "are you done testing?" to "is the level of risk understood and accepted?"

Gartner

How does this business-focused approach to Continuous Testing work? At a high level, you situate a broad set of automated defect prevention and detection practices that serve as "sensors" throughout the SDLC— continuously measuring both the product and the process. If the product falls short of expectations, you don't just remove the problems from the faulty product. You also consider each problem found an opportunity to re-examine and optimize the process itself—including the effectiveness of your sensors. This establishes a defect-prevention feedback loop that enables you to incrementally improve the process.

In terms of DevOps, the benefits of Continuous Testing include:

- Business stakeholders always have real-time access to feedback on whether their expectations are being met, enabling them to make informed decisions.
- At the time of the critical "go/no go" decision, there is an objective assessment of whether the organization's specific expectations are satisfied—reducing the business risk of a fully-automated Continuous Delivery process.
- Defects are eliminated at the point when they are easiest, fastest, and least costly to fix—a prime principle of being "lean."
- Continuous measurement vs. key metrics means continuous feedback, which can be shared and used to refine the process.

This notion of "continuous quality" is central to achieving the expected ROI from DevOps, agile, and other lean initiatives. We hope that you find this collection of resources helpful as you consider how to transform your SDLC to achieve the optimal balance of quality and speed in this new era of "Continuous Everything."

Source: Parasoft

What's Needed for Continuous Testing?

Consider this: if software quality has traditionally been a "time-boxed" exercise, then we can't possibly expect that accelerating the SDLC will yield better results from a testing perspective. If organizations want to accelerate software releases, they must reassess the current testing practices in order to keep quality as status quo. However, in order to improve software quality in conjunction with SDLC acceleration, organizations will have to truly consider re-engineering the software quality process.

As you begin the transformation to Continuous Testing, the following elements are necessary for achieving a real-time assessment of business risks.

Risk Assessment—Are You Ready to Release?

As we review the elements of Continuous Testing, it's hard to argue that one element is more important than the rest. If we present our case well enough, it should become obvious that each element is critical for overall process success. However, we need a place to start— and establishing a baseline to measure risk is the perfect place to begin as well as end.

One overarching aspect to risk assessment associated with software development is continuously overlooked: If software is the interface to your business, then developers writing and testing code are making business decisions on behalf of the business.

FIGURE 1 Elements of Continuous Testing

Source: Parasoft

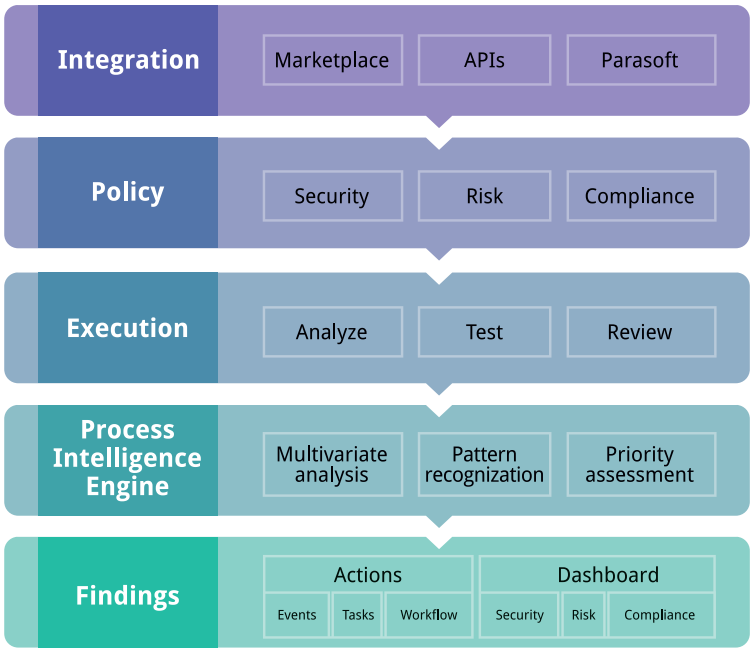
What is a Development Testing Platform?

This level of automation required for Continuous Testing requires a method to federate quality information from multiple infrastructure sources (source code management, build management, defect management, testing, etc.). A Development Testing Platform is this central "system of decision" which translates policies into prioritized tasks as well as delivers insight and control over the process of creating quality software.

To truly optimize the SDLC, we need to move away from the old concept of tool selection and into the concept of process enablement. One of the biggest contributors to the lack of process consistency in the SDLC is the ad-hoc nature in which tools are adopted and deployed. We're not saying that the organization must standardize on a single tool or brand—in fact, we're suggesting quite the opposite. The best tools must be adopted in order to achieve the optimal business outcome. However, those tools must be managed within the context of established business expectations (policy) in order to provide uniform analysis, consistent reporting, and measurable outcomes.

If speed is the primary definition for team success, quality will inevitable suffer unless you have established expectations that are automatically monitored for compliance. Making quality expectations non-negotiable sets the boundaries for acceleration while reducing the risks associated with application or project failure. In other words, a Development Testing Platform assists the organization to work smarter—limiting the nature and degree in which business-critical tasks can be discounted.

FIGURE 2 Development Testing Platform



Source: Parasoft

Assessing the project risk upfront should be the baseline by which we measure whether we are done testing and allow the SDLC to continue towards release. Furthermore, the risk assessment will also play an important role in improvement initiatives for subsequent development cycles.

The definition of risk cannot be generic. It must be relative to the business, the project, and potentially the iterations in scope for the release candidate. For example, a non-critical internal application would not face the same level of scrutiny as a publically-exposed application that manages financial or retail transactions. A company baseline policy for expectations around security, reliability, performance, maintainability, availability, legal, etc. is recommended as the minimum starting point for any development effort. However, each specific project team should augment the baseline requirement with additional policies to prevent threats that could be unique to the project team, application, or release.

SDLC acceleration requires automation. Automation requires machine-readable instructions which allow for the execution of prescribed actions (at a specific point in time). The more metadata that a team can provide around the application, components, requirements, and tasks associated with the release, the more rigorous downstream activities can be performed for defect prevention, test construction, test execution, and maintenance.

Technical Debt

Core to the management of SDLC risk is the identification of technical debt. A Development Testing Platform will help prevent and mitigate types of technical debt such as poorly-written code, overly-complex code, obsolete code, unused code, duplicate code, code not covered by automated tests, and incomplete code. The uniform measurement of technical debt is a great tool for project comparison and should be a core element of a senior manager's dashboard.

Risk Mitigation Tasks

All quality tasks requested of development should be 100% correlated to a defect prevention policy or an opportunity to minimize risk or technical debt. A developer has two primary jobs: implement business requirements and reduce the business risk associated with application failure. From a quality and testing perspective, it is crucial to realize that quality initiatives generally fail when the benefits associated with a testing task are not clearly understood.

A risk mitigation task can range from executing a peer code review to constructing or maintaining a component test. Whether a risk mitigation task is generated manually at the request of a manager or automatically (as with static code analysis), it must present a development or testing activity that is clearly correlated with the reduction of risk.

Coverage Optimization

Coverage is always a contentious topic—and, at times, a religious war. Different coverage techniques are better-suited for different risk mitigation goals. Fortunately, industry compliance guidelines are available to help you determine which coverage metric or technique to select and standardize around.

Once a coverage technique (line, statement, function, modified condition, decision, path, etc.) is selected and correlated to a testing practice, the Development Testing Platform will generate reports as well as tasks that guide the developer or tester to optimize coverage. The trick with this analysis is to optimize versus two goals. First, if there is a non-negotiable industry standard, optimize based on what's needed for compliance. Second (and orthogonal to the first), optimize on what's needed to reduce business risks.

Coverage analysis is tricky because it is not guaranteed to yield better quality. Yet, coverage analysis can certainly help you make prioritization decisions associated with test resource allocation.

Test Quality Assessment

Processes and test suites have one thing in common: over time, they grow in size and complexity until they reach a breaking point when they are deemed "unmanageable." Unfortunately, test suite rationalization is traditionally managed as a batch process between releases. Managing in this manner yields to sub-optimal decisions because the team is forced to wrangle with requirements, functions, or code when the critical details are no longer fresh in their minds.

Continuous Testing requires reliable, trustworthy tests. When test suite results become questionable, there is a rapid decline in how and when team members react. This leads to the test suite becoming out-of-sync with the code—and ultimately out of control.

With this in mind, it is just as important to assess the quality of the test itself as it is to respond to a failing test. Automating the assessment of the test is critical for Continuous Testing. Tests lie at the core of software risk assessment. If these risk monitors are not reliable, then we must consider the process to be out of control.

Policy Analysis—Keep up with Evolving Business Demands

Policy analysis through a Development Testing Platform is key for driving development and testing process outcomes. The primary goal of process analysis to ensure that policies are meeting the organization's evolving business and compliance demands.

Most organizations have a development or SDLC policy that is passive and reactive. This policy might be referenced when a new hire is brought onboard or when some drastic incident compels management to consult, update, and train on the policy. The reactive nature of how management expectations are expressed and measured poses a significant business risk. The lack of a coordinated governance mechanism also severely hampers IT productivity (since you can't improve what you can't measure).

Policy analysis through a Development Testing Platform is the solution to this pervasive issue. With a central interface where a manager or group lead defines and implements "how," "when," and "why" quality practices are implemented and enforced, management can adapt the process to evolving market conditions, changing regulatory environments, or customer demands. The result: management goals and expectations are translated into executable and monitor-able actions.

The primary business objectives of policy analysis are:

- Expose trends associated with the injection of dangerous patterns in the code
- Target areas where risks can be isolated within a stage
- Identify higher risk activities where defect prevention practices need to be augmented or applied

With effective policy analysis, "policy" is no longer relegated to being a reactive measure that documents what is assumed to occur; it is promoted to being the primary driver for risk mitigation.

As IT deliverables increasingly serve as the "face" of the business, the inherent risks associated with application failure expose the organization to severe financial repercussions. Furthermore, business stakeholders are demanding increased visibility into corporate governance mechanisms. This means that merely documenting policies and processes is no longer sufficient; we must also demonstrate that policies are actually executed in practice.

This centralization of management expectations not only establishes the reference point needed to analyze risk, but also provides the control required to continuously improve the process of delivering software.

Requirements Traceability—Defending the Business Objective

All tests should be correlated with a business requirement. This **provides an objective assessment of which requirements are working as expected, which require validation**, and which are at risk. This is tricky because the articulation of a requirement, the generation or validation of code, and the generation of a test that validates its proper implementation all require human interaction. We must have ways to ensure that the artifacts are aligned with the true business objective—and this requires human review and endorsement. Continuous Testing must promote the validation of testing artifacts via peer review.

A Development Testing Platform helps the organization keep business expectations in check by ensuring that there are *effective* tests aligned to the business requirement. By allowing extended metadata to be associated with a requirement, an application, a component, or iteration, the Development Testing Platform will also optimize the prioritization of tasks.

During "change time," continuous tests are what trigger alerts to the project team about changes that impact business requirements, test suites, and peripheral application components. In addition to satisfying compliance mandates, such as safety-critical, automotive, or medical device standards, real-time visibility into the quality status of each requirement helps to prevent late-cycle surprises that threaten to derail schedules and/or place approval in jeopardy.

Advanced Analysis—Expose Application Risks Early

Defect Prevention with Static Analysis

It's well known that the later in the development process a defect is found, the more difficult, costly, and time-consuming it is to remove. Mature static analysis technologies, managed in context of defined business objectives, will significantly improve software quality by preventing defects early.

Writing code without static code analysis is like writing a term paper or producing a report without spell check or grammar check. A surprising number of high-risk software defects are 100% preventable via fully-automated static code analysis.
By preventing defects from being introduced in the first place, you minimize the number of interruptions and delays caused by the team having to diagnose and repair errors. Moreover, the more defects you prevent, the lower your risk of defects slipping through your testing procedures and making their way to the end-user—and requiring a significant amount of resources for defect reproduction, defect remediation, re-testing, and releasing the updated application. *Ultimately, automated defect prevention practices increase velocity, allowing the team to accomplish more within an iteration.*

At a more technical level, this automated analysis for defect prevention can involve a number of technologies, including multivariate analysis that exposes malicious patterns in the code, areas of high risk, and/or areas more vulnerable to risk. All are driven by a policy that defines how code should be written and tested to satisfy the organization's expectations in terms of security, reliability, performance, and compliance. The findings from this analysis establish a baseline that can be used as a basis for continuous improvement.

Pure "defect prevention" approaches can eliminate defects that result in crashes, deadlocks, erratic behavior, and performance degradation. A security-focused approach can apply the same preventative strategy to security vulnerabilities, preventing input-based attacks, backdoor vulnerabilities, weak security controls, exposure of sensitive data, and more.

Change Impact Analysis

It is well known that defects are more likely to be introduced when modifying code associated with older, more complex code bases. In fact, a recent FDA study of medical device recalls found that an astonishing "192 (or 79%) [of software-related recalls] were caused by software defects that were introduced when changes were made to the software after its initial production and distribution."[1]

From a risk perspective, changed code equates to risky code. We know that when code changes, there are distinct impacts from a testing perspective:

- Do I need to modify or eliminate the old test?
- Do I need a new test?
- How have changes impacted other aspects of the application?

The goal is to have a single view of the change impacts from the perspective of the project as well as the perspective of the individual contributor. Optimally, change impact analysis is performed as close to the time of change as possible—when the code and associated requirements are still fresh in the developer's or tester's mind.

If test assets are not aligned with the actual business requirements, then Continuous Testing will quickly become unmanageable. Teams will need to spend considerable time sorting through reported failures—or worse, overlook defects that would have been exposed by a more accurate test construction.

Now that development processes are increasingly iterative (more agile), keeping automated tests and associated test environments in sync with continuously-evolving system dependencies can consume considerable resources. To mitigate this challenge, it's helpful to have a fast, easy, and accurate way of updating test assets. This requires methods to assess how change impacts existing artifacts as well as a means to quickly update those artifacts to reflect the current business requirements.

Scope and Prioritization

Given a software project's scope, iteration, or release, some tests are certainly more valuable and timely than others. Advanced analysis techniques can help teams identify untested requirements, tasks, and code. Advanced analysis should also deliver a prioritized list of regression tests that need review or maintenance.

Leveraging this type of analysis and acting on the prioritized test creation or maintenance tasks can effectively prevent defects from propagating to downstream processes, where defect detection is more difficult and expensive. There are two main drivers for the delivery of tasks here: the boundaries for scope and the policy that defines the business risks associated with the application.

For example, the team might be working on a composite application in which one component is designed to collect and process payment cards for online transactions. The cost of quality associated with this component can be colossal if the organization has a security breach or fails a PCI DSS[2] audit. Although code within the online transaction component might not be changing, test metadata associated with the component could place it in scope for testing. Furthermore, a policy defined for the PCI DSS standard (as well as the organization's internal data privacy and security) will drive the scope of testing practices associated with this release or iteration.

Test Optimization—Ensure Findings are Accurate and Actionable

To truly accelerate the SDLC, we have to look at testing much differently. In most industries, modern quality processes are focused on optimizing the process with the goal of preventing defects or containing defects within a specific stage. With software development, we have shied away from this approach, declaring that it would impede engineering creativity or that the benefits associated with the activity are low, given the value of the engineering resources. With a reassessment of the true cost of software quality, many organizations will have to make major cultural changes to combat the higher penalties for faulty software. Older, more established organizations will also need to keep up with the new breed of businesses that were conceived with software as their core competency. These businesses are free from older cultural paradigms that might preclude more modern software quality processes and testing practices.

No matter what methodology is the best fit for your business objectives and desired development culture, a process to drive consistency is required for long-term success.

Test optimization algorithms help you determine what tests you absolutely must run versus what tests are of lower priority given the scope of change. Ideally, you want intelligent guidance on the most efficient way to mitigate the greatest risks associated with your application. Test optimization not only ensures that the test suite is validating the correct application behavior, but also assesses each test itself for effectiveness and maintainability.

Management

Test optimization management requires that a uniform workflow is established and maintained associated with the policies defined at the beginning of a project or iteration. A Development Testing Platform must provide the granular management of queues combined with task workflow and measurement of compliance. To achieve this:

- The scope of prescribed tasks should be measurable at different levels of granularity, including individual, team, iteration, and project.
- The test execution queues should allow for the prioritization of test runs based on the severity and business risk associated with requirements.
- Task queues should be visible and prioritized with the option to manually alter or prioritize (this should be the exception, not the norm).
- Reports on aged tasks should be available for managers to help them determine whether the process is under control or out of control.

Construction

With a fragile test suite, Continuous Testing just isn't feasible. If you truly want to automate the execution of a broad test suite—embracing unit, component, integration, functional, performance, and security testing—you need to ensure that your test suite is up to the task. How do you achieve this? Ensure that your tests are...

- **Logically-componentized:** Tests need to be logically-componentized so you can assess the impact at change time. When tests fail and they're logically correlated to components, it is much easier to establish priority and associate tasks to the correct resource.
- **Incremental:** Tests can be built upon each other, without impacting the integrity of the original or new test case.
- **Repeatable:** Tests can be executed over and over again with each incremental build, integration, or release process.
- **Deterministic and meaningful:** Tests must be clean and deterministic. Pass and fail have unambiguous meanings. Each test should do exactly what you want it to do—no more and no less. Tests should fail only when an actual problem you care about has been detected. Moreover, the failure should be obvious and clearly communicate what went wrong.
- **Maintainable within a process:** A test that's out of sync with the code will either generate incorrect failures (false positives) or overlook real problems (false negatives). An automated process for evolving test artifacts is just as important as the construction of new tests.
- **Prescriptive workflow based on results:** When a test does fail, it should trigger a process-driven workflow that lets team members know what's expected and how to proceed. This typically includes a prioritized task list.

Test Data Management

Access to realistic test data can significantly increase the effectiveness of a test suite. Good test data and test data management practices will increase coverage as well as drive more accurate results. However, developing or accessing test data can be a considerable challenge—in terms of time, effort, and compliance. Copying production data can be risky (and potentially illegal). Asking database administrators to provide the necessary data is typically fraught with delays. Moreover, delegating this task to dev/QA moves team members beyond their core competencies, potentially delaying other aspects of the project for what might be imprecise or incomplete results.

Thus, fast and easy access to realistic test data removes a significant roadblock. The primary methods to derive test data are:

- Sub-set or copy data from a production database into a staged environment and employ cleansing techniques to eliminate data privacy or security risks.

- Leverage Service Virtualization (discussed later in this resource) to capture request and response traffic and reuse the data for subsequent scenarios. Depending on the origin and condition of the data, cleansing techniques might be required.

- Generate test data synthetically for various scenarios that are required for testing.

In all cases, it's critical to ensure that the data can be reused and shared across multiple teams, projects, versions, and releases. Reuse of "safe" test data can significantly increase the speed of test construction, management, and maintenance.

Maintenance

All too often, we find development teams carving out time between releases in order to "clean-up" the test suites. This ad-hoc task is usually a low priority and gets deferred by high-urgency customer feature requests, field defects, and other business imperatives. The resulting lack of ongoing maintenance typically ends up eroding the team's confidence in the test suite and spawning a backlog of increasingly-complex maintenance decisions.

Test maintenance should be performed as soon as possible after a new business requirement is implemented (or, in the case of TDD-like methodologies, prior to a requirement being implemented). The challenge is to achieve the optimal balance between creating and maintaining test suites versus the scope of change.

Out-of-sync test suites enter into a vicious downward spiral that accelerates with time. Unit, component, and integration tests that are maintained by developers are traditionally the artifacts at greatest risk of deterioration. Advanced analysis of the test artifact itself should guide developers to maintain the test suite. There are five primary activities for maintenance—all of which are driven by the business requirement:

- Delete the test
- Update the test
- Update the assertions
- Update the test data
- Update the test metadata

[1] http://www.fda.gov/medicaldevices/deviceregulationandguidance/

[2] PCI DSS is the Payment Card Industry Data Security Standard

Service Virtualization—Eliminate Test Environment Access Issues

With the convergent trends of parallel development and increasing system complexity/interdependency, it has become extremely rare for a team to have ubiquitous access to all of the dependent applications required to execute a complete test. By leveraging Service Virtualization to remove these constraints, an organization can gain full access to (and control over) the test environment—enabling Continuous Testing to occur as early and often as needed.

Want to start testing the component you just built even though not much else is completed? Don't have 24/7 access to all the dependencies involved in your testing efforts—with all the configurations you need to feel confident that your test results are truly predictive of real-world behavior? Tired of delaying performance testing because access to a realistic environment is too limited (or too expensive)? Service Virtualization can remove all these constraints.

With Service Virtualization, organizations can access simulated test environments that allow developers, QA, and performance testers to test earlier, faster, and more completely. Organizations that rely on interconnected systems must be able to validate system changes more effectively—not only for performance and reliability, but also to reduce risks associated with security, privacy, and business interruption. Service Virtualization is the missing link that allows organizations to continuously test and validate business requirements. Ultimately, Service Virtualization brings higher quality functionality to the market faster and at a lower cost.

Source: Parasoft

Research from Gartner

Market Trends: DevOps — Not a Market, but a Tool-Centric Philosophy That Supports a Continuous Delivery Value Chain

Bimodal and digital business strategies stimulate demand for improved speed and effectiveness of software delivery. Product management should focus on tool functionality that supports agile methodologies, automation and collaborative relationships to assist customers in achieving a DevOps goal.

Key Findings

- DevOps is a cultural shift that merges operations with development and demands a linked toolchain of technologies to facilitate collaborative change
- Interchangeability is key to the success of the DevOps toolchain since each deployment is different and can require a unique combination of tools to support key functions — loosely coupled via APIs rather than the heavily integrated/hardwired, difficult-to-maintain tools available today.
- Lack of needed functionality will result in the swapping of one solution (tool) for another, forcing vendors to be constantly vigilant. However, such flexibility can add to solution fragility and support costs.
- OSS tool adoption and appetite remain strong; however, large-enterprise clients prefer commercially supported OSS distributions.

Recommendations

- Develop and link tools that address multiple facets of the DevOps philosophy together in a toolchain to provide the overall benefits.
- Prepare and execute varied strategies to address the needs of a variety of buyers, including development, test, operations, cloud and application organizations.
- Match evolution of each offering to the changing requirements of the DevOps movement, either through development or acquisition.
- Implement a try-before-you-buy strategy for new tool offerings; OSS solutions have been successful with this strategy when upgrades to commercial offerings are seamless.

Strategic Planning Assumption

By 2016, DevOps will evolve from a niche strategy employed by large cloud providers to a mainstream strategy employed by 25% of Global 2000 organizations.

Introduction

This document was revised on 25 February 2015. The document you are viewing is the corrected version. For more information, see the Corrections page on gartner.com.

Gartner believes that DevOps is a philosophy (not a market). There are no rules or manuals, only guidelines; therefore, adoption and implementation will vary greatly. Its foundation remains focused on the adoption of agile and lean methodologies and a collaborative relationship between development (Dev) and operations (Ops), with a singular goal of a timely, successful application production rollout.

The DevOps philosophy finds traction among three significantly different groups of adopters. Each of these groups is seeking to overcome time constraints and resource requirements, while ensuring the quality and stability of their releases.

- The original group of agile practitioners seeks to match the delivery and deployment pace to that of development teams using agile methods. The release pace ranges from daily to monthly.
- In the second group are Web-scale practitioners, who adapt DevOps notions but also have restructured the runtime architectures and deployment mechanisms, managing deployment and risk in novel ways to achieve many microreleases in a day.

- Finally, organizations that have well-established structured release practices (often regulated entities, such as financial institutions) are embracing aspects of DevOps. In this last group, the release pace may currently be much slower than in the other two, but there are still demands for more frequent release or more efficient release at the appropriate level of quality.

To provide enhanced guidance to both technology suppliers as well as end-user consumers, Gartner has elected to discuss DevOps as a virtual (and likely temporal) market. Because the potential definition is very broad, Gartner has focused the scope of this virtual market definition on the tools that support DevOps and practices associated with it in the context of continuous delivery, continuous improvement, infrastructure and configuration as code, and so on. This methodology focuses on building a toolchain of loosely coupled tools to address continuous integration and delivery.

This toolchain includes not only the tools that support a continuous movement into production, but also the tools that provide information on the health and performance of the latest release as a continuation of the agile feedback loop. DevOps tools considered for this virtual market must support the characteristics and functionality traits listed in Table 1.

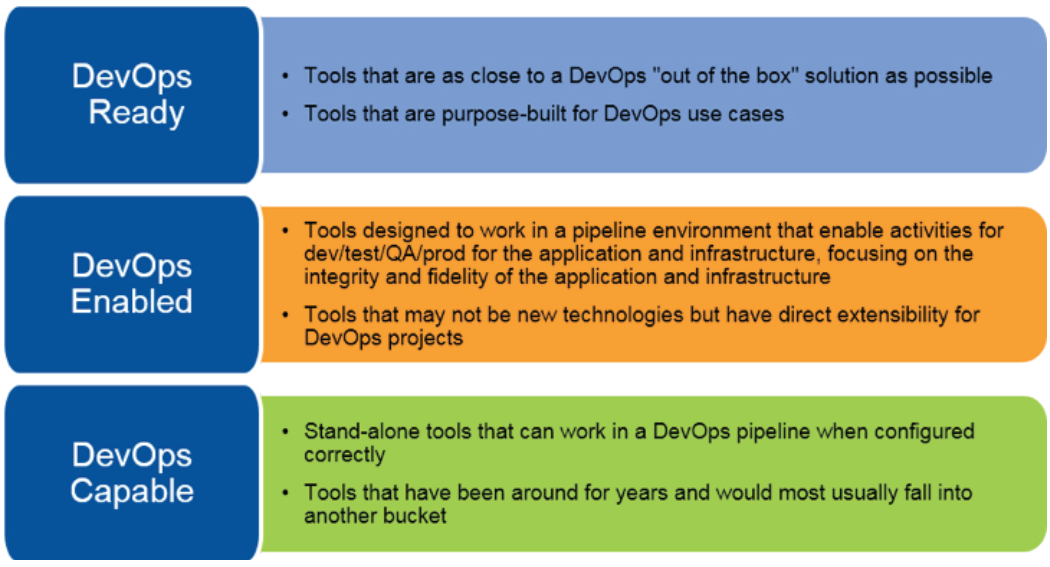
Table 1. DevOps Tool Characteristics and Traits

Characteristics and Traits	
Agile	Customizable
Automated	Extensible
Collaborative	Modern
Composiable — command line	Specifiable
Contextual	Web user interface — scriptable
Continuous	Workflows

Source: Gartner (February 2015)

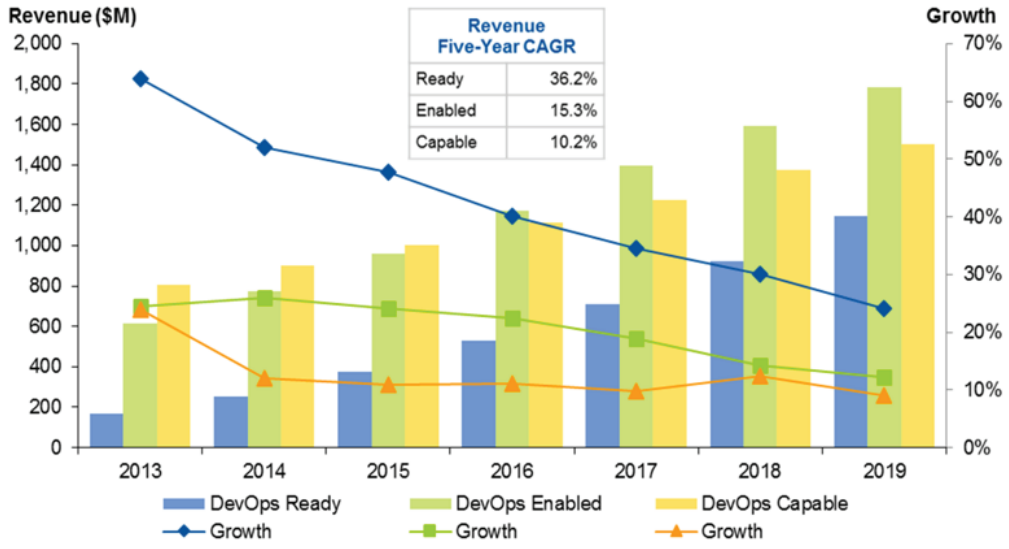
Although DevOps emphasizes people (and culture) over tools and processes, implementations utilize technology, especially automation tools that can leverage an increasingly programmable and dynamic infrastructure from a life cycle perspective. Gartner categorizes these tools as DevOps-ready, -enabled, and -capable tools. Table 1 details the tool qualifications necessary for inclusion as a DevOps support tool, while Figure 1 provides a description of each category.

FIGURE 1 DevOps Tool Categorization



dev/test/QA/prod = development, test, quality assurance and production
Source: Gartner (February 2015)

FIGURE 2 Estimated DevOps Market Size and Growth for DevOps-Ready, DevOps-Enabled and DevOps-Capable Tools



CAGR = compound annual growth rate
Note: DevOps is a composite market, comprising software tools that are part of other major markets within Gartner Market Share and Forecast documents.
Source: Gartner (February 2015)

In the response to the desire to deliver IT services by leveraging lean and agile practices, the tools required to aid the necessary cultural shift are rapidly becoming the increased focus for traditional IT operations and development tool vendors, in addition to startups and open-source projects designed specifically for DevOps support. As a result, Gartner expects strong growth opportunities for the three DevOps toolsets (ready, enabled and capable), as described in Figure 1.

Figure 2 provides a composite market view in terms of size and projected growth of the DevOps toolchain in terms of total software revenue. This composite view was estimated based on revenue accounted for in Gartner Market Share and Forecast reports for application development (AD) and IT operations management (ITOM) tools that fit the criteria defined and outlined in this document. See the DevOps Tools and Vendors to Watch section for an outline of the product types that make up each of the three DevOps tool categories. Not all products in corresponding sections of the reported AD and ITOM market subcategories fit into the DevOps tool definitions; therefore, only a percentage of each apply to the view provided in Figure 2.

Predictably, DevOps-ready tools have seen and will continue to see the largest growth potential. These tools are specifically designed and built with out-of-the-box functionality to support the described DevOps characteristics and traits. Most DevOps-enabled and-capable tools currently exist as part of the larger IT operation and development toolbox; however, with time to value as a critical demand factor from clients, emphasis in support of DevOps has transformed how these tools are positioned and perceived in the marketplace.

Market Trend

Digital Business Makes Adoption of DevOps Practices Essential

In response to the rapid change in business today, DevOps can help organizations that are pushing to implement a bimodal strategy to support their digitalization efforts. Digital business is software — this means that organizations that expect to thrive in a digital business environment must have an improved competence in software delivery.

A leaner and increasingly mobile workforce requires a different approach to development, testing, management and deployment of

applications; however, automated practices and a "team effort method" for agile IT are necessary to maintain some level of cross-corporate consistency.

As end-user buying power shifts away from central IT to line-of-business and other organizational managers, demand for speed, tool intuitiveness and software effectiveness from nontechnical users will increase significantly. In addition, the expanding use cases for mobile devices and Web services with an ever increasing number of innovative applications and even higher customer (consumerlike) expectations add to DevOps growth in support of consistent deployment and updating of applications. This trend goes beyond implementation and technology management and instead necessitates a deeper focus on how to effect positive organizational change.

DevOps Philosophy Shapes a Culture

The DevOps philosophy was born primarily from the activities of cloud service (and Web 2.0) providers as they addressed scale-out problems due to increasing online service adoption. Constant change and complexity necessitated developing improved effectiveness through collaboration between operations and development. Rethinking organizational structure of the IT department has become paramount: especially rethinking the value of people who understand multiple functional perspectives, such as development, test and operations, rather than just development. Likewise, as operations teams leverage new tools to codify the infrastructure programmatically, operations teams are viewed in the mode of "system developers" and adopt some practices of professional programming organizations. This shift brings credibility with operations peer group (developers), which helps bring these teams together more collaboratively.

As a result of these changes, vendors with tools that support and enable organizational change become more relevant. Both closed-source and commercial open-source software (OSS) vendors are now shifting focus to offerings that support this movement. As the philosophy continues to evolve, it will be introduced to traditional enterprises that struggle to manage increasing business demand and rapid change brought about by digital business and by bimodal strategies. Similarly, as developers and operations staff build DevOps teams, tools can be acquired and leveraged across both teams (the unified team); therefore, it is critical for vendors to build solutions that can be leveraged in development, test and/or production. This positioning will enable vendors to find multiple buyers and extend sales opportunities as DevOps projects grow.

Influencers of Change

The DevOps philosophy centers on people, process, technology and information. With respect to culture, DevOps seeks to change the dynamics in which operations and development teams interact. Key to this change are the issues of trust, honesty and responsibility. In essence, the goal is to enable each organization to see the perspective of the other and to modify behavior accordingly, while motivating autonomy without the "hero attitude." However, people are only a portion of the equation; continual improvement of the right processes and accurate information at the right time are also necessary to optimize value.

Technology providers need to focus on developing tools to accommodate the business need for change and collaboration. This demand creates an opportunity for vendors to provide tools that enable toolchains with open programmatic APIs. Even if a vendor does not provide the framework for a toolchain, all tools "earmarked" for DevOps must be developed with the foresight that they are part of a bigger toolchain ecosystem.

The DevOps movement is influenced by the following:

- Faster and more effective application changes to meet growing bimodal delivery options in support of digitalization requires IT to find a new way of doing things.
- Differences in scope and granularity and/or accuracy of information lead to conflicting definitions and acceptable tolerance of risk (meaning things are built to break — and recover).
- Many existing tools are not designed to span both development and production requirements in a single toolchain to meet the needs of engineering and operations and DevOps teams.
- Process improvement initiatives are forcing coordination among competing processes, with emphasis on both efficiency and effectiveness.

Foremost, DevOps is about changing culture, which necessitates a different set of measurements and incentives to be effective. There is no one single way to achieve this change. Instead, Gartner encourages IT organizations to develop shared metrics for engineering and operations or designed for a DevOps team to ensure that everyone is focused on the same goal. Senior management needs to look for methods that foster innovative thinking and responsibility. Most importantly with respect to DevOps, CIOs and other IT leaders need to sanction risk taking that does not result in damage to one's career. To support such change, these organizations will require an effective toolchain to guide them on a journey of transformation.

DevOps Trends

The overall DevOps message is compelling, because many enterprise IT organizations want to achieve the scale-out and economies of scale achieved by world-class cloud providers. However, there are still several gaps that prevent implementation of DevOps as a comprehensive methodology.

The first area needing more definition is IT operations processes. While DevOps focuses intensively on release and configuration management and, to a lesser extent, on monitoring, it has limited focus on other key operations process areas. There is also little to say with regard to organizational issues beyond the notion for improved collaboration between development and operations. For example, there are no new roles or organizational styles suggested, even though improvisations abound.

Enterprises have acknowledged the gaps and have begun assessing how the DevOps mindset might apply to their own environments. However, culture is not easily or quickly changed. And key to the culture within DevOps is the notion of becoming more agile and changing behavior to support it — a perspective that has not been widely pursued within classical IT operations. IT cultures are characteristically centrally controlled and independently siloed, leading little in the way of responsibility (in fact, today's cultures are heavily focused on blame). In addition, IT leadership needs to change as well to create an environment that incubates a behavior of respect and disciplined freedom.

One of the biggest issues with DevOps is that each implementation is unique. No two are the same because there is no prescribed method to implement. This fact makes it difficult for organizational success and harder for vendor positioning since it is not a predictable market. Supporting the DevOps toolchain opens a wide opportunity for a vendor with the sales savvy to target a variety of buyers from multiple organizations, including development, test, operations, cloud and applications.

DevOps Tools

DevOps philosophies emerged as organizations tried to address speed and scaling limitations that slowed the deployment of systems of innovation into production. Lean and agile principles pointed to simplification or standardization to remove sources of variation, provide quick feedback and corrective response to failures or errors, and provide higher levels of automation and use of configuration as code in support of both these ends. Open-source projects developed much of the early tooling in applying these principles.

Subsequently, OSS vendors began introducing commercial tools that focused on infrastructure as code. Over time, these tools have evolved and they will continue to do so. For example, application release automation (ARA) tools started out moving only application code, and then environment context was added to the products, and most recently process design and collaborative workflow (such as release coordination). Although many OSS configuration automation tools are not new, the timing of DevOps projects, the push from developers to have operations be more engineering-focused and the recognition of the credibility of OSS tools for production management (not just for developers) have all helped raise the visibility of and shift to commercial versions.

One way to look at tooling within DevOps is as a component within a DevOps toolchain. New sets of tools are emerging that address management functions (such as monitoring and continuous build, integration and testing) specific to the DevOps philosophy. However, toolchains are not about these individual categories of tools; their importance resides in the ability of tools (development, testing, management and deployment) to plug together and the ease of which new ones can be substituted for old ones

(swapped out and plugged in). This plug-and-play concept is radically different from traditional management tools that require more of an "all (of one vendor) or nothing" model. Even so, traditional AD and ITOM tool vendors are joining a growing number of providers (OSS and startups) that are addressing and extending solutions beyond the initial orientation in support of DevOps.

An organization cannot simply buy one tool or even multiple tools to gain the advantages of a DevOps approach. Changes in philosophy, practice and tooling are all necessary to navigate the transformation.

The Future of DevOps

Innovations in DevOps continue to emerge in the Web-scale and agile development communities. Some form of DevOps is implicit in all Web-scale deployments, but not all the implementations can extend generally to all situations. Organizations seeking to reach Web-scale must re-engineer their runtime, deployment, monitoring and risk management to incorporate DevOps ideas and practices. Current efforts often require custom implementation. We expect open-source and commercial tools to increasingly address these needs.

Organizations with agile development will be slower to embrace DevOps across the entire application life cycle. Cultural resistance and low levels of process discipline will create significant failure rates for DevOps initiatives, particularly when waterfall processes are still a dominant portion of the development portfolio. We expect a majority of enterprises attempting to scale agile over the next five years to recognize the need for DevOps initiatives.

Organizations with structured release practices will more slowly adopt some DevOps tooling but not always acknowledge the philosophy. Among these buyers, DevOps offers the most advantage when both engineering and operations are working together (instead of one or the other driving the entire project).

Multiple paces and targets have to be reconciled during release. An example would be a coordinated release of Web front-end code, custom Java application server code, and back-end package or mainframe code. Long term, five years or more out, we expect audit and compliance standards to embrace and require at least DevOps approaches for regulated or material applications.

Why DevOps strategies and tools will continue to be adopted:

- The DevOps principles of reducing sources of variation and increasing automation are strongly embraced by many existing operational philosophies.
- DevOps can be driven as either a top-down or bottom-up strategy; however, a bottom-up strategy is often more easily accepted by IT operations teams that had a hand in strategy development.
- Enterprise IT is increasingly mindful of the efficiencies and agility obtained by large cloud providers and their use of DevOps.
- IT organizations are seeking to ways to improve where ITIL and other best-practice framework initiatives have not delivered on their goals.
- The growing interest in continuous configuration automation tools by both developers and operations will help stimulate demand for OSS-based management.
- Development of Web and mobile applications that require continuous improvements and enhancements will drive the need for tools that address those specific needs.

The next five years will see increased demand from organizations for tools that support toolchains designed to ease the facilitation of DevOps without bogging down change with volumes of processes. References and case studies that detail avenues for change will be most useful for promoting DevOps solutions.

Contrarian View

Sales for DevOps support tools could lose traction if the DevOps initiatives become tooling exercises that neglect process discipline and transformation. Changes in behavior are difficult to cultivate and usually take longer than expected. The fact that DevOps adoption varies so greatly by organization (and most projects are small) creates a big challenge for vendors to find the right buyer and to acquire revenue (at least initially) — also, toolchains imply that no vendor can rest on its laurels. Interchangeability implies any tool can be pulled out for a different one (albeit this is unproved thus far). In light of these identified difficulties, Gartner has included a few scenarios that could occur to stagnate or inhibit projected growth of this virtual market.

Will DevOps Become a Fizzled Overhyped Word Rather Than an Opportunity for Change?

Influencers that could inhibit DevOps adoption:

- DevOps requires acceptance of significant changes in IT operations organization, practices and culture by both IT workers and management.
- Current investments in infrastructure, applications and people are insufficient to successfully implement this approach.
- Organizations ignore changing needs and cling to the large body of work with respect to ITIL and other best-practice frameworks that are already accepted within the industry.
- The lack of prescriptive routes to DevOps implementation creates the perception that it is radical, unreliable or high-risk.
- Without a definitive "framework, template, formula" to guide implementation, adoption (and subsequently success) may be difficult to demonstrate widely.
- OSS management tools, which are more aligned with this approach, fail to gain significant enterprise market share traction.
- Operations staff are unable to develop or acquire either the skills necessary to adopt philosophy or tools that support it.

Vendors to Watch

Sample Vendors

This research does not constitute an exhaustive list of vendors in any given technology area, but rather it is designed to highlight interesting, new and/or innovative vendors, products and services. Gartner disclaims all warranties, express or implied, with respect to this research, including any warranties of merchantability or fitness for a particular purpose.

DevOps-Ready Tools

DevOps-ready tools are out-of-the-box solutions designed to support at least one of the workflow steps in DevOps. Tools included in this category are ARA and continuous configuration automation software.

ARA tools offer automation to enable best practices in moving related artifacts, applications, configurations and even data together across the application life cycle. To do so, ARA tools provide a combination of automation, environment modeling and workflow management capabilities to simultaneously improve the quality and velocity of application releases. These tools are a key part of enabling the DevOps goal of achieving continuous delivery with large numbers of rapid small releases.
Continuous configuration automation tools provide a programmatic platform to codify various activities predominantly focused on configuring systems. The platform leverages a proprietary coding language (but leverages content from open-source communities). Both traditional vendors as well as startups and commercial OSS projects are focused on developing (or acquiring) functionality to support these tools. Table 2 lists vendors that offer DevOps-ready tools (it is a sampling and not an exhaustive list).

Table 2. Sample List of Vendors With DevOps-Ready Tools

Application Release Automation	Continuous Configuration Automation
BMC	Ansible
CA Technologies	CFEngine
Electric Cloud	Chef
IBM	Puppet Labs
MidVision	SaltStack
Serena Software	
VMware	
XebiaLabs	

Source: Gartner (February 2015)

DevOps-Enabled Tools

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, focusing 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 include continuous integration, continuous quality, code review and static analysis (static application security testing [SAST]), testing automation, and environment (pipeline) management tools.

Contitnuous integration tools monitor source repositories and then carry out preconfigured sequences of action whenever new code is checked in. These actions include ensuring pre-check-in conditions are met, calling the compiler and executing quality functions, such as static analysis and build verification testing.

Continuous quality tools include service/data/network virtualization — lab provisioning/management, code review, static analysis, unit test/smoke testing — automated test tools. The extension from agile and continuous integration to continuous delivery and utilization of DevOps concepts relies on the ability to drive continuous quality. The goal is to shift quality left and drive faster delivery of functionality. This requires a higher degree of automation and a more productionlike test environment. In addition, agile has driven in a number of additional practices that push quality through the process, such as test-driven development, peer code reviews and static analysis on every build.

SAST is a set of technologies designed to analyze application source code, bytecode and binaries for coding and design conditions that are indicative of security vulnerabilities. SAST solutions analyze an application from the "inside out" in a nonrunning state.

Testing automation tools enables the definition, construction, automation, cataloging and execution of suites of test cases. Tools may highlight new or changed outcomes and provide rapid feedback of test results to designated developers or other staff.

Environment management tools enable modeling, provisioning and configuration of environments. These tools can provide one, two or all three functions. Environments can include but are not limited to infrastructure as a service (IaaS), private platform as a service and application environments. These tools are usually focused on private on-premises environments, but they can also manage environments in the public cloud, as well.

Table 3 lists vendors that offer DevOps-enabled tools (it is a sampling and not an exhaustive list).

Table 3. Sample List of Vendors With DevOps-Enabled Tools

Continuous Integration		Continuous Quality		Code Review and Static Analysis		Test Automation		Environment Management	
Atlassian	Jenkins	CA Technologies	Kubysis	Atlassian	Optimyth	Borland	QASymphony	Actifio	Delphix
CircleCI	JetBrains		Mockito	Cast	Parasoft	Concordion	SmartBear Software	Atlassian	Docker
Codenvy	Microsoft	Delphix	Parasoft	Checkstyle	SonarQube	Cucumber		BMC	IBM
Electric Cloud	OpenMake Software	HP	Toptal	Coverity	Sonatype	FitNesse	TestPlant	CloudBees	Jenkins
Hudson		IBM	Tricentis	FindBugs	Veracode	IBM	TestRail	CoreOS (Rocket)	OpenMake Software
IncrediBuild	Travis CI			Gerrit	Virtual Forge	Microsoft	Zephyr	ElasticBox	Red Hat
				Microsoft	WhiteHat Security	Sauce Labs		Electric Cloud	VMware
				Omnext				DBmaestro	Cloud management platform vendors (that do more than IaaS)

Source: Gartner (February 2015)

Table 4. Sample List of Vendors With DevOps-Capable Tools

Monitoring		Security Testing	Lab Management
AppDynamics	Ganglia	Acunetix	Amazon
AppFirst	Graphite	HP	Citrix
AppNeta	Graylog2	IBM	CloudBees
Caliper	Librato	N-Stalker	Codenvy
Circonus	New Boundary Technologies	NT OBJECTives	CollabNet
collectd	New Relic	PortSwigger	Dell (Quest Software)
Compuware	Reconnoiter	Qualys	HP
Datadog	SAP (OpTier)	Trend Micro	IBM
Elasticsearch/ Logstash/ Kibana	Splunk	Trustwave	Microsoft
	Sumo Logic	Veracode	Ravello Systems
		WhiteHat Security	Skytap
			VMware

Source: Gartner (February 2015)

DevOps-Capable Tools

DevOps capable tools have (typically) been around for years, but they usually fall into another ITOM category. However, these stand-alone tools can work in a DevOps pipeline when configured correctly. Monitoring, security testing (dynamic application security testing [DAST]) 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.

Monitoring tools live above the operating system layer, focusing on the application metrics and being able to ingest and analyze custom metrics often generated by the application code. They do include information supplied by other infrastructure components for additional context. The focus of these tools is to collect, analyze and alert on end-user experience, release quality and custom business metrics.

DAST technologies are designed to detect conditions indicative of a security vulnerability in an application in its running state. Most DAST solutions test only the exposed HTTP and HTML interfaces of Web-enabled applications; however, some solutions are designed specifically for non-Web protocol and data malformation (for example, remote procedure call, SIP and so on).

Lab management tools manage the provisioning, operation, archiving and retirement of virtual test facilities in support of multiple teams, projects and organizations.

Evidence

This report required the collection of data and the preparation of market statistics. We have taken into account the following when doing our analysis: prevailing market conditions and political and economic events that affect vendor performance (such as regulations, mergers and acquisitions, a slow worldwide economic recovery, and migration to new versions of software).

Gartner uses public sources of information and works with software vendors to establish estimates for market sizing. Information from Gartner's secondary research and internal community meetings has also been used to arrive at certain conclusions. The data in this research report is published as Gartner estimates/opinion, and not as facts that vendors report.

Source: Gartner Research, G00274555,
Laurie Wurster, Ronni Colville, Jim Duggan,
18 February 2015

About Parasoft

Parasoft researches and develops software solutions that help organizations deliver defect-free software efficiently. By integrating development testing, API testing, and service virtualization, we reduce the time, effort, and cost of delivering secure, reliable, and compliant software. Parasoft's enterprise and embedded development solutions are the industry's most comprehensive—including static analysis, unit testing, requirements traceability, coverage analysis, functional and load testing, dev/test environment management, 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.

Development Testing Platform

Parasoft Development Testing Platform (DTP) eliminates the business risk of faulty software by consistently applying software quality practices throughout the SDLC. Parasoft DTP enables your software quality efforts to shift left, delivering a platform for automated defect prevention and the uniform measurement of risk across project teams. With seamless integration into any SDLC infrastructure system, including open source and third-party testing tools, Parasoft DTP allows you to aggregate disparate data and apply statistical analysis techniques—transforming traditional reporting into a central system of decision.

API/Integration Testing

As the risks associated with application failure have broader business impacts, the integrity of the APIs you produce and consume is now more important than ever. Parasoft's API Testing solution provides centralized visibility and control of services and APIs—ensuring security, reliability, and performance.

Service Virtualization

With Parasoft's Service Virtualization solution, organizations bring higher quality software to the market faster and at a lower cost. Parasoft's industry-leading test environment simulation and management technologies enable developers, QA, and performance testers to test earlier, faster, and more completely.