PARASOFT®

WHITEPAPER

Develop a Strategy and Business Case for Test Automation

USING ROI CALCULATIONS TO DERIVE VALUE

THE COST OF POOR SOFTWARE QUALITY

What is the business case for improving software quality? There are obvious savings in reduced fixes and lower support costs that come with most quality improvements. However, the real payoff is in improved customer retention and acquisition with, presumably, market share growth.

To begin a discussion on a business case and the value of software quality, it's important to establish the cost of poor software quality. Software is expensive to create, maintain, and support. Inevitably, poor quality software increases the costs of all these dimensions.

It turns out that poor software quality is very expensive. According to the report from the Consortium for Information & Software Quality (CISQ), The Cost of Poor Software Quality in the US: A 2020 Report, the total cost was estimated at $2.08 trillion, approximately 10% of the US GDP for 2020. The largest contributor to this number is the "meteoric rise of cybersecurity failures" with an "underlying cause is primarily unmitigated flaws in software."

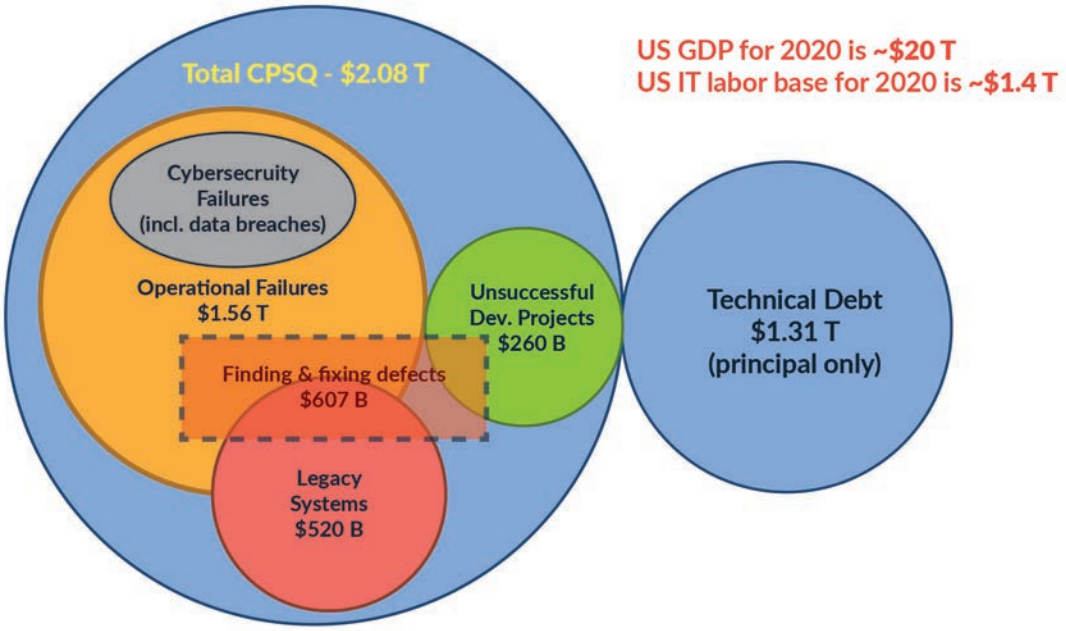


Figure 1:
Source: The Cost of Poor Software Quality in the US: A 2020 Report, from the Consortium for Information & Software Quality

THE IMPACT OF POOR SOFTWARE QUALITY

The place where poor software quality impacts the bottom line the most is when it ruins customer experiences. Many companies recognize this and place a higher importance on customer experience as it relates to the software aspects of their brand. Corporations specifically call out software quality as a potential risk to their business in their annual reports and at investor conferences.

Figure 2: 2020/2019 Annual Reports & Investor Conferences—quality at the executive level.

This acknowledgement is a step in the right direction, but as the CISQ report shows, many companies are struggling with customer experience and software quality.

One of Parasoft's customers, Caesars Entertainment, considers guest experience their number one priority and is making strides in improving software quality as part of this goal.

Along the way, Caesars and others face challenges in their journey to improving software quality. A large part of this is testing as it remains the best way we can verify and validate software.

Improving quality implies improving testing. As large organizations like Caesars understand, manual testing doesn't scale, and with an expanding portfolio of organizations, they needed to make sure they could meet their customer experience goals. Take a look at Caesars' journey and how they, and others, determine the value of test automation using return on investment (ROI) calculations.

SETTING TEST AUTOMATION PRIORITIES

To improve the automation of a testing process, it's important to prioritize its focus. This is driven by where test automation provides the most value.

Every organization has its "problem children." These are areas of the application heavily used by customers and where most of the bug reports come from. It makes sense to concentrate the automation in these areas first. But as the test practice matures, the goal to increase coverage means that other areas of the application are tested by automation as well.

MANUAL TESTING REMAINS IMPORTANT

Test automation isn't meant to replace manual testing altogether, but rather reduce the mundane testing to let the team focus on usability and customer experience. Teams need to weigh which tests make sense to automate.

Manual testing remains important for both user experience and exploratory testing. Human testers are much better at determining that the implemented functionality fulfills its intention. These tests have limited up-front costs and are easy to maintain. However, manual tests are expensive and time consuming to do, and extensive regression testing is prohibitive within Agile sprints.

Figure 3: Manual or automated testing? Teams need to do both!

On the other hand, automated tests have an up-front cost and little to no ongoing costs. When done right, they are reliable and repeatable and take much less time than manual tests.

The correct balance is a combination of both with the emphasis for manual tests to be on usability, UX, and exploratory testing. Test automation excels at removing tedious and repetitive tasks from human testers. It's also highly repeatable and, importantly, scalable.

BUILDING A SCALABLE TEST AUTOMATION STRATEGY

Consider the concept of the test pyramid, which is well-known to Agile practitioners. The ideal state to establish an efficient and scalable strategy is to allocate testing and developer resources in a way that critical portions of the application are tested as early as possible. The idea is to spend more time and effort on building quality into the application with deep code analysis, unit testing, and API testing. End-to-end testing of the application is minimized and focused on the scenarios that are critical to validate the customer experience.

It's important to do both manual and automated testing of the UI. Automated testing exposes regressions in existing functionality, while manual testing focuses on the usability of new capabilities.

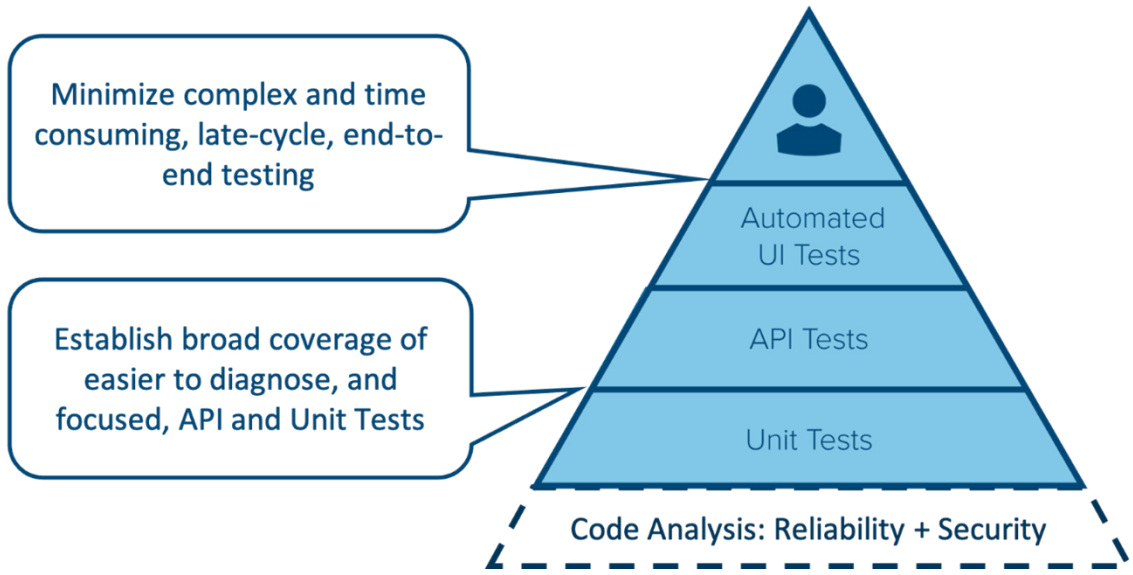


Figure 4: Building a scalable test automation strategy.

A successful test strategy relies on building on a foundation of testing smaller scale components and moving up to the other levels (units to APIs, APIs to UIs). It's likely your current strategy isn't quite there yet. If it is, hopefully you are reaping the benefits. If not, then let's examine the evolution required.

Using a bottom-up approach, consider the following recommendations at each level of the pyramid.

Automated unit testing is a proven technique for ensuring software quality. While it might seem like a burden to developers, the payoff is efficient execution with quick feedback if changes have impacted the application.

In order to get the most out of testing and automated testing tools, tests must be trustworthy, maintainable, readable, self-contained, and be used to verify a single use case. Automation is key to making unit testing workable and scalable.

Automated API Testing provides an ideal way to validate the integrated business logic and serves as a communication mechanism between developers and testers. With a high level of maintainable automation, API testing can extend into, and shift left, performance and security testing.

API testing decouples the underlying complexity of modern enterprise applications. When used in conjunction with virtualized services, it provides an efficient way to test business logic without the need for time-consuming and brittle UI tests. Shifting these tests left and executing them earlier in the software life cycle means catching critical security and architectural defects early, where they are easier to diagnose and less risky to fix.

Manual and automated UI testing are still critical but when the full foundation of the test pyramid supports it, there's more time for exploratory and in-depth UX testing.

Although the original test pyramid starts at unit testing, code quality and security need to be assured before submitting to the repository and unit testing. Static analysis is a proven way to improve code quality and security. It's critical in enforcing project-wide coding standards. Detecting bugs and vulnerabilities before the code is tested decreases testing effort and reduces the chances of serious bugs getting through the testing process.

BEST PRACTICES FOR DEFINING A SCALABLE TEST AUTOMATION PRACTICE

Improving a test automation strategy doesn't happen overnight. It requires investment and commitment. Here are some best practices to help improve an organization's strategy.

Invest in People

It's not necessary to hire new people if you can retain and train your existing team. Those on the team that are motivated to take on automation should be trained on the tools and techniques that will benefit the business. Specialists can be used during initial stages to help kick start the initiative and help establish test practices and processes.

Define the Process

It's critical that test automation goals are aligned with product quality and business goals. Metrics and reporting need to be aligned with these goals so the team understands what the goals and the quality status are at any time.

Consolidate Technology

Technology shouldn't lead the test automation strategy but, rather, be defined by the goals and requirements. Often the toolchain consists of open source frameworks and commercial tools.

To facilitate the scaling of test automation, it's important to define a toolchain that is consolidated and consistent across the organization. It's also critical to consider the long-term goals, not just the short-term needs, when selecting tools.

THE VALUE OF TEST AUTOMATION EVOLVES WITH TEST STRATEGY

Establishing a scalable test solution in a real-world organization is an evolution that takes several years. There's no silver bullet solution that migrates an enterprise software organization from a manual-heavy testing approach to a balanced, automated one overnight. Take, for example, the case of Caesars Entertainment and their seven year journey to improving their testing approach.

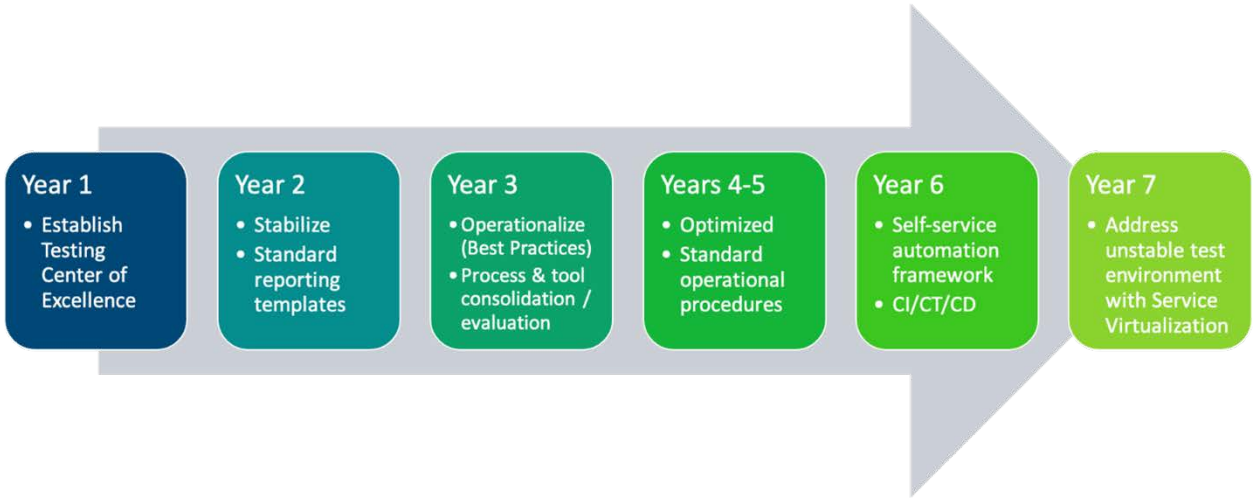


Figure 5: Caesars Entertainment's seven year testing journey to quality management.

In their **first year**, Caesars established a foundation to build on. In their case, it was a center of excellence for software testing. The name doesn't matter. What's important is the commitment to improving quality and testing standards and practices.

By **year two**, they had stabilized their testing practice and established a standardized set of reports to help communicate test and product status.

In **year three**, the testing practice was spreading across the enterprise along with standardized processes and tools.

By **years four and five**, Caesars was optimizing their test frameworks and standardized operational procedures. By **year six**, they were integrating continuous testing into continuous integration/delivery (CI/CD) pipelines.

In **year seven**, they expanded their test practice with service virtualization to improve their API test strategy.

Although quality improvement is a long-term commitment, improving an organization's test automation strategy is a journey with worthwhile rewards.

When advocating for quality and transformation of existing processes, whether testing or otherwise, you must establish an understanding why it's important. The keys to obtaining buy-in to this commitment to improvement is to understand:

- How automation helps.
- Why you need it.
- What's the expected ROI for every dollar invested in new practices and tools?

It's also essential to be able to communicate this value to management. Since explaining the value is so important, how is it measured? The common way to do this is by comparing the costs of implementing new tools and techniques versus the savings they bring about—the ROI.

MEASURING THE ROI OF TEST AUTOMATION

COST AVOIDANCE METHOD

One way to measure the ROI of adopting a new process or technology is by tracking how much time has been saved compared to previous projects or time periods. This method isn't tracking individual defects and estimating costs, it's tracking the actual change in behavior over time. As efficiencies improve with automation, so does the cost associated with the entire project.

CASE STUDY

Caesars tracked their ROI using a cost avoidance method associated with the automation of manual activities (that is, how much did they save by accelerating their testing with automation). As their journey from early adoption on test projects moved to large scale projects, the ROI grew over time as their test strategy evolved.

The formula they used for ROI considers the manual effort traditionally required for testing and effort invested in test automation—test creation and maintenance and the execution time. Coming from a mostly manual process where MT would be large, this customer decreased the overall time spent on testing significantly as their adoption matured.

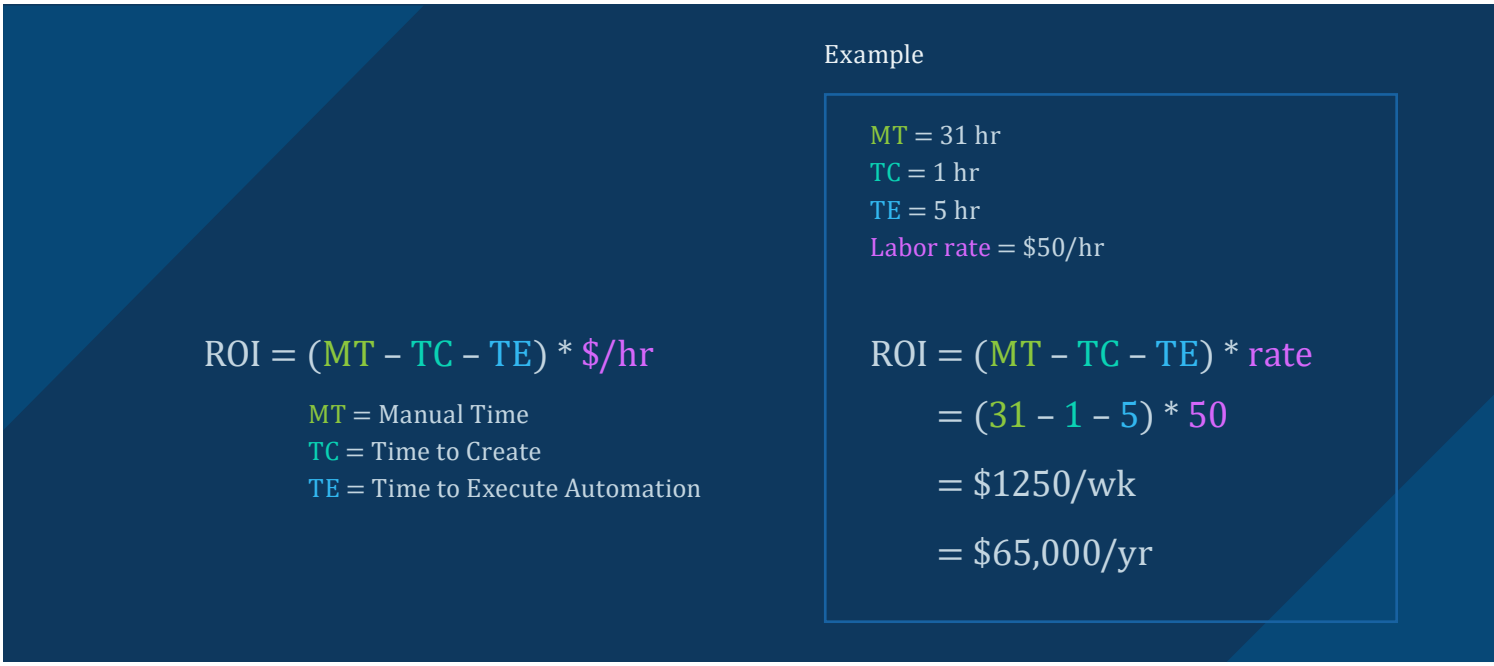


Figure 6: Caesars Entertainment's formula for tracking the ROI of new processes or technologies.

The following graph shows Caesars' savings over time. For their calculations they used a labor rate of $50/hr. By year six, they were seeing savings over $850K per year and by year 7 and beyond, they projected over $1 million in savings.

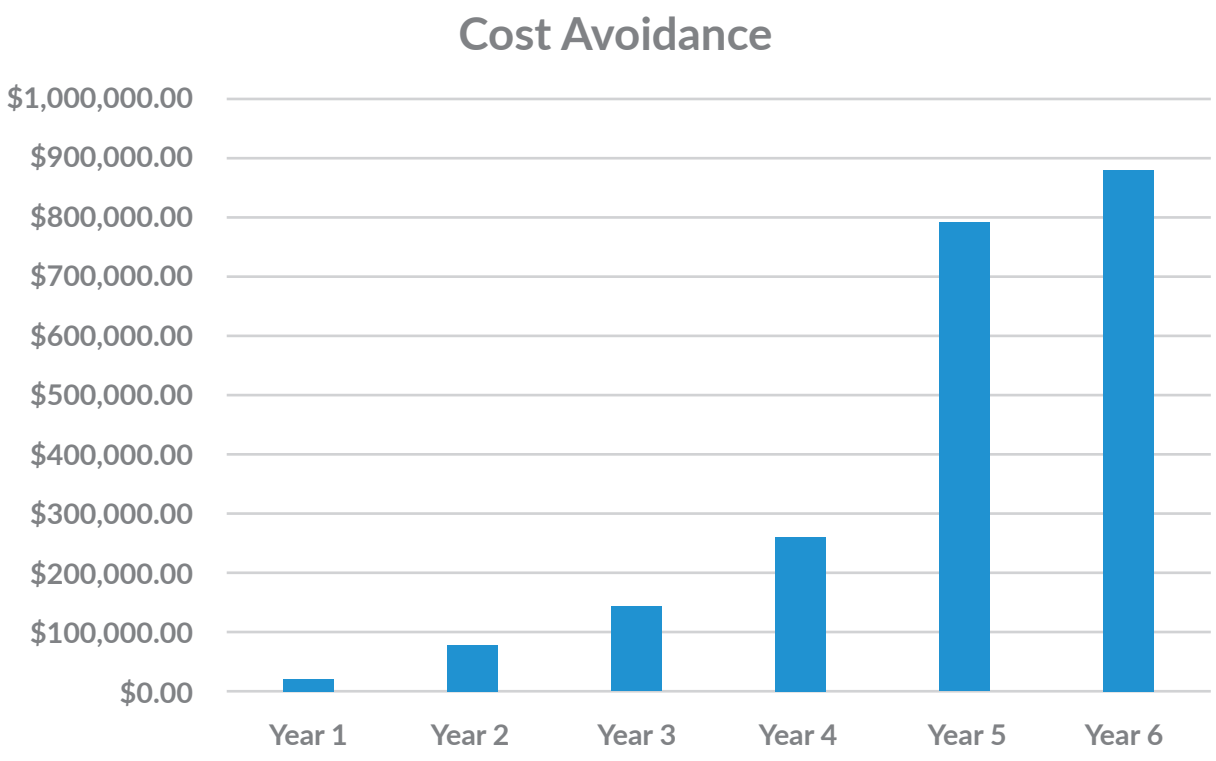


Figure 7: Caesars Entertainment's savings over a six year span of time.

COST SAVINGS FROM EARLY DEFECT REMOVAL

Another common way to measure the ROI of new tools and processes is to calculate how much a discovered, prevented, or avoided defect or security vulnerability would cost if it survived unscathed through the development process and into customer hands.

It's easy and cheap to fix bugs during the coding phase but unfortunately most bugs aren't found until much later in the development lifecycle. See the defects introduced versus defects found in the following graph.

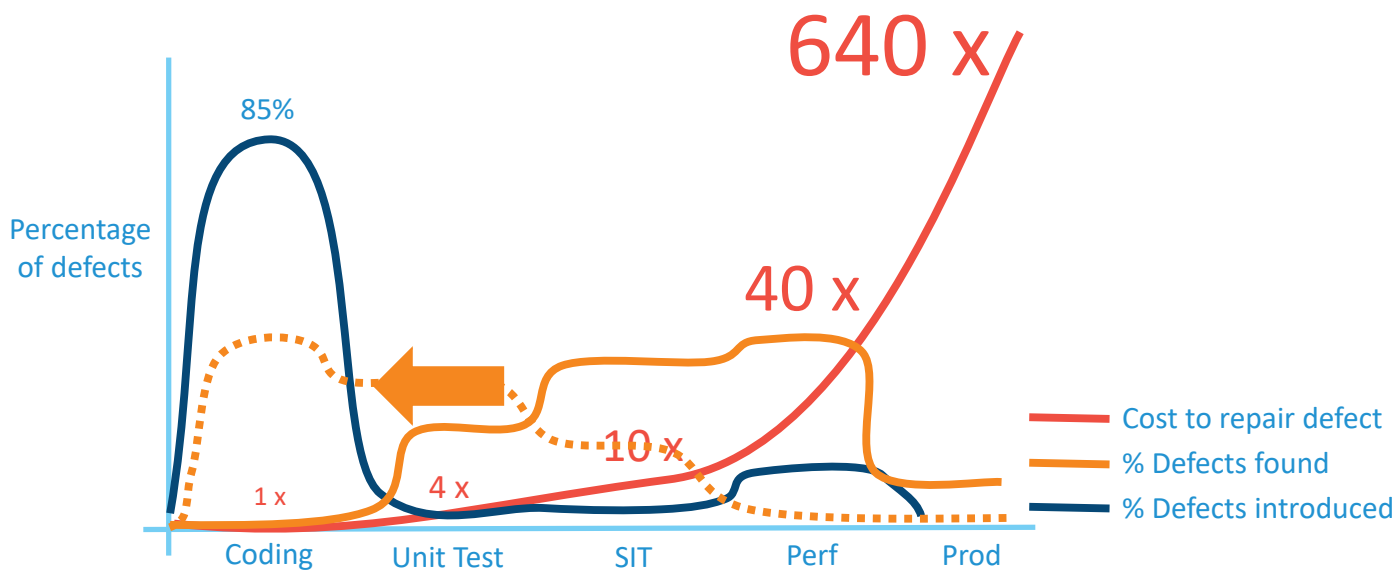


Figure 8: The reduced cost of defects when detected early or prevented altogether.

Finding and fixing bugs after the coding phase gets more and more expensive and the cost to fix goes up exponentially. The business driver behind "shift left" is related to this graph. Finding and fixing defects early is cheaper and easier, and improves outcomes for downstream processes.

As the yellow curve illustrates, moving the defects found curve to the left reduces the defects found both early and late in the cycle with a significant impact on costs and schedule.

Capers Jones is a well-known researcher in this area and keeps an up-to-date study on the effect of defect density and defect removal efficiency (DRE) and its impact on software quality. Put simply, DRE can be used to measure the effect on quality from adopting a new process or technology.

If DRE increases, then the new practice is improving quality. However, Jones is quick to point out that there's more to return on investment for increased quality than the money saved fixing a defect earlier. While it's a valuable metric, it actually undersells the ROI and does not properly correlate to actual software quality.

CALCULATING ROI FROM COST PER DEFECT

Using only the cost-per-defect metric and approach to calculating ROI, consider the example above with a team of ten people working on a project with a loaded labor rate of $100 per hour.

This team, using new test automation tools with all the benefits (shifting left the identification of defects), discovers 20 more defects than they did in the previous sprints. Finding and fixing these bugs early might require three hours per defect for a total of $6,000.

This ROI calculation depends on knowing how much a defect costs to fix later on in the development cycle. Estimates are often used based on the graph above showing the multiplier at each stage of development. Later is more expensive.

For the sake of this example, finding and fixing these bugs later in integration or system testing might triple the effort, costing $18,000 for these 20 defects.

$$ROI = DR * (CPD - (ATF * LLR))$$

DR – defects removed
CPD – full cost of defect if it made it to final stages of the project
ATF – average time to fix
LLR – loaded labor rate

Figure 9: Calculation of ROI from cost per defect.

Example

DR = 20 defects
CPD = $900
ATF = 3 hours
LLR = $100

$$ROI = DR * (CPD - (ATF * LLR))$$
$$= 20 * ($900 - (3 * $100))$$
$$= $12,000$$

Figure 10: Simplistic example of ROI calculation.

Why Cost Per Defect Undersells the ROI

Simplistically, for the example above, the ROI is $12,000. Sounds great, right? This is actually a conservative estimate.

The example doesn't factor in the two days of development time savings or an extra sprint that was avoided due to these early fixes. Factoring in the impact early defect removal has on the project is trickier to account for because it might be hard to estimate how many sprints were added due to accumulated defects that had to be fixed. However, the ROI calculation needs to include some factor of the lost time and effort that is pre-emptively prevented.

NES – number of extra, unplanned sprints
CPS – cost per sprint

$$ROI = DR * (CPD - (ATF * LLR)) + NES * CPS$$

Figure 11: ROI calculation that factors in lost time and effort that's prevented pre-emptively.

Just as an example, a team of ten people cost $8,000 per day at $100/hour. Two lost days due to these 20 defects (not an unreasonable amount) is an additional $16,000.

One might be quick to point out that it doesn't take ten people to fix these errors, which is correct, of course. However, an extra sprint costs money, and the entire team is still being paid regardless of where the setback lies.

> "However, when economic analysis includes the savings associated with shorter schedules, it will be seen that the economic value of quality is directly proportional to the size of the application measured with function points. The larger the application, the more valuable high quality becomes. This phenomenon cannot be measured using cost per defect, but it can be measured using economic analysis based on total application schedules and costs."
>
> — Capers Jones, A Short History of the Cost Per Defect Metric

In the context of communicating ROI of test automation tools, it's important to look at the larger picture and the savings from the overall improvement in schedules that comes with better quality. With larger teams and larger projects come larger savings and higher ROI on tools. Which leads to a more direct ROI method: accounting for the savings directly.

HOW PARASOFT HELPS DELIVER THE VALUE OF TEST AUTOMATION

There is a role to play in the shift to automated testing. Although key to making the process more efficient, the transition still requires commitment to the processes and culture change to be successful. However, the right tools selection can help software teams realize the benefits of this transformation easier and sooner.

The following section discusses various Parasoft tools and how they contribute to the value of test automation. Each tool applies to a different layer of the test pyramid as illustrated below.

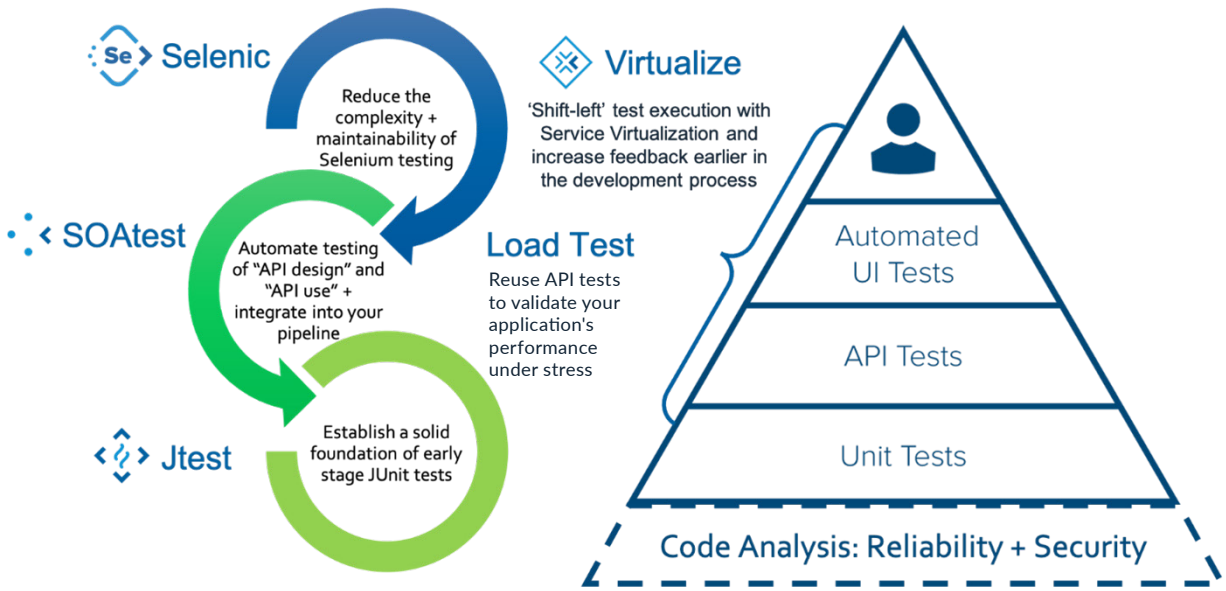


Figure 12: The test pyramid and how each Parasoft tool applies to the different layers.

SHIFT DEFECT AND SECURITY VULNERABILITY DETECTION TO WHERE CODE IS CREATED

Parasoft Jtest verifies Java code quality and checks compliance with security standards (OWASP, CWE, PCI DSS, and more) by applying a wide range of static analysis checkers and using the most comprehensive set of static code analyzers to go way beyond open source. Parasoft Jtest leverages Parasoft's centralized reporting and analytics hub, Parasoft DTP, with its Process Intelligence Engine to provide deep insights on code quality and risk.

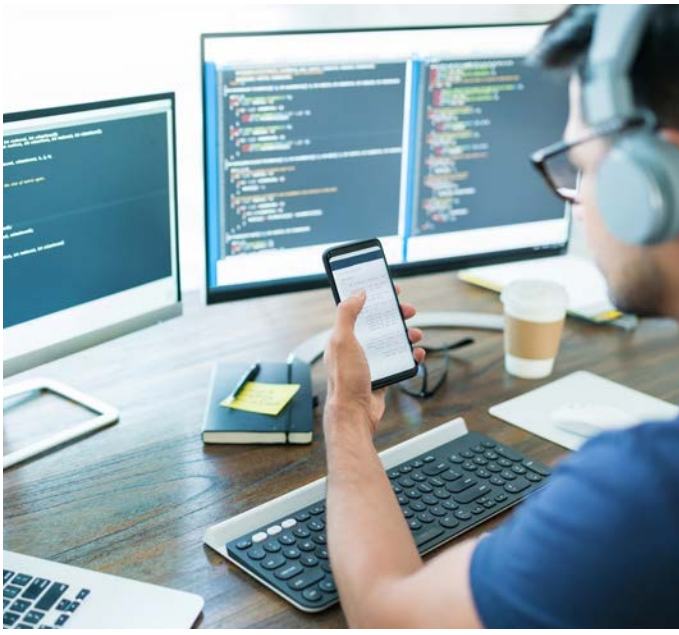
Parasoft Jtest plugs into both the CI/CD pipeline and the developer's IDE (Eclipse, IntelliJ) where it automatically analyzes the code in the background and alerts developers when it detects issues, giving immediate feedback as early as possible, effectively shifting quality and security left to the point code is created. In addition to reviewing static analysis results directly in the IDE, results are also available as HTML, PDF, and custom extension reports.

One big benefit from embedding static code analysis into the software development process is that of the business intelligence of the current state of the product and key indicators of risk, enabling software teams to focus on key areas of their product. Without this ability, users must assemble multiple reporting products and integrate each individual tool.

GREATLY REDUCE UNIT TEST CREATION, EXECUTION, & MAINTENANCE

Parasoft Jtest provides users with AI assistance to help make unit testing easier and faster. Users can easily generate, augment, and reuse unit tests, while optimizing the execution of those tests, to substantially reduce the amount of time and cost associated with unit testing. Jtest users report creating JUnit test suites twice as fast, meeting coverage targets while building a comprehensive, meaningful, and maintainable suite of JUnit test cases, and releasing with confidence.

Jtest identifies areas of untested code and provides recommendations for how to cover it. Then, with quick-fix actions, Jtest can clone and mutate existing tests to cover the untested code. Jtest's one-of-a-kind bulk generation capability enables users to create unit tests en masse for an entire project, package, or class, automatically performing all the groundwork, set up, and even mocking, giving the user a jump start on their unit test suite. Users simply add in additional business logic to the tests and execute them.



Parasoft Jtest makes maintaining unit tests more automated, reducing the burden of upkeep. Jtest automatically determines when assertions are no longer valid and provides quick fixes to replace the code with correct assertions. Jtest also helps identify instabilities in the test environment and provides recommendations on how to resolve these issues.

With its AI capabilities, Jtest correlates source code changes in the development environment with the entire JUnit test suite and performs test impact analysis. This benefits users by enabling them to automatically execute only the unit tests that were affected by the correlated source code changes, rather than having to run the entire suite of tests repeatedly. This workflow can be used in the IDE as well as the CI process.

Parasoft Jtest enables users to control the coverage collecting process in real time (for manual or automated functional tests) when coverage data is being collected. Users can start or stop test sessions and download current coverage data to correlate coverage information with the test and the person doing the test.

REDUCE RELIANCE ON MANUAL UI TESTING BY GROWING YOUR API TESTING PRACTICE

Parasoft SOAtest delivers a fully integrated set of API and web service testing tools that automate end-to-end functional API testing. Streamline automated functional testing of business logic with advanced codeless test creation capabilities for applications with multiple interfaces (REST & SOAP APIs, microservices, databases, and more). The tools reduce the risk of security breaches and performance outages by transforming functional testing artifacts into security and load equivalents. Such reuse, along with continuous monitoring of APIs for change, allows faster and more efficient testing.

SOAtest provides a single, intuitive interface and automates complex testing scenarios for 120+ message formats and protocols, covering microservices to mainframes. Testers gain confidence knowing that change impact analysis continuously monitors APIs, highlights changes and corresponding test cases for updates, and streamlines test refactoring efforts. As is critical in CI/CD pipelines, SOAtest gives immediate and intelligent feedback for smarter API testing and on-time releases.

To relieve the burden of manual testing at the UI level, API testing is made easier and more efficient to adopt with Parasoft SOAtest. Automatic creation of functional tests from recorded traffic is a critical time saver: API calls from an application's web interface can be captured directly in the Chrome web browser, using SOAtest's smart API test generator plugin to build tests, leveraging advanced heuristics and artificial intelligence to produce meaningful and complex test scenarios. Machine learning is employed to learn about the underlying business logic from any test in your test library, enabling SOAtest to intelligently create or update any of your test assets in exact accordance with how your business has decided to test that API.

As API testing strategy scales, libraries of test cases grow, and when the APIs being tested change, tests need to be updated. Ordinarily this causes a significant barrier to scaling the automation strategy, but SOAtest automates API change management. Parasoft SOAtest's Change Advisor proactively scans API interfaces, looks for changes in the services, then identifies how the test assets are impacted by those changes and helps users easily update them.

Executing complete test suites for every incremental build is very time consuming and becomes a bottleneck in every CI/CD pipeline. Instead of executing all the tests to verify the quality of a build, SOAtest optimizes API test suites to execute only the tests necessary to validate the changes between builds. Within its smart test execution capabilities, Parasoft SOAtest uses test impact analysis to optimize the set of tests to be executed so you can get quicker feedback from the CI/CD pipeline.

To enable visibility and tracking of functional test results, Parasoft SOAtest generates HTML reports and XML output with results that can be published into continuous integration systems as well as to Parasoft's centralized reporting server for additional reporting and analytics.

OFFLOAD MANUAL UI TESTING TO STREAMLINED SELENIUM TESTING

Selenium is one of the most popular tools for UI testing. However, testing with Selenium requires confidence that failures are due to UI errors rather than failures in the tests themselves. While Selenium tests are brittle in the face of constant UI changes, it offers many benefits. Parasoft Selenic applies AI heuristics to determine if failures are due to a real regression in the application. This self-healing capability uses enhanced locator and wait condition strategies to detect unstable tests, modify them on the fly during execution, and then illustrate where the changes were made.

Focus on real issues instead of wasting cycles on failed test runs. Parasoft Selenic not only highlights test execution issues, it also provides recommendations for fixes to the tests that will make them more reliable. Locator recommendations can be imported directly into the integrated development environment for one-click test updates that ensure successful testing in the future.

Parasoft Selenic's Recorder captures UI actions within the Chrome browser. The Recorder also defines locators with elements specific to Salesforce, Guidewire, and other enterprise applications. After the capture, use the recording to create easily maintainable pure Java-based Selenium tests with assertions, built using the Page Object Model for maximum maintainability.

Manual UI testing is very time consuming and slow compared to automated testing. Parasoft Selenic and Selenium together accelerate automated UI testing. In particular, Selenic optimizes your Selenium test suite to execute only the tests required to validate code changes between builds. More efficient testing delivers faster feedback. Less testing takes less time, providing an ever larger benefit multiplier versus manual testing time.

REMOVE BARRIERS TO TESTING BY REDUCING ENVIRONMENT CHALLENGES

To realize the benefits from Agile and DevOps initiatives, teams need instant access to their test environment, free of constraints. By applying service virtualization in testing environments, organizations can reduce or eliminate the reliance on unavailable, unstable, or costly dependencies, such as 3rd party services, databases, and mainframes. Parasoft Virtualize's intuitive service virtualization solution makes it easy for users to create, scale, and share virtual services.

Parasoft Virtualize easily creates complex test conditions (such as "what-if," security, failover, performance, and negative test scenarios), and uncovers hidden performance issues in your application under test by controlling the performance of the service, for load testing or slow network simulation. Users can easily configure a variety of scenarios by dynamically data-driving service virtualization from external data sources, including Parasoft's powerful data repository infrastructure.

One of the largest challenges to realistic service virtualization is test data management. Parasoft's test data solution doesn't require users to learn a brand-new discipline. Test data is created from the previously recorded API traffic used to create virtual services. Parasoft Virtualize can store and manage all the data in the proprietary data repository system. This lightweight data storage mechanism makes it much easier to load and utilize data without the tedious effort of creating SQL queries.

By building a smaller virtual service from the beginning and then loading it with all the necessary data by generating synthetic data, it's much easier to use and maintain virtual services while reducing maintenance. Parasoft Virtualize provides a snapshot for the "golden state" of the test data, which, when altered through usage, can be reset right back to its original starting point with a series of simple API calls.

OTHER CONSIDERATIONS FOR IMPLEMENTING A SUCCESSFUL TEST STRATEGY

To achieve a successful transformation of an organization's quality processes, there are some key considerations needed in addition to the calculation and communication of ROI.

- **Leadership support** is critical in order to obtain (and retain) the motivation, investment, and direction needed to make quality improvement a priority and lead the required software development culture change.
- **Culture change** is inevitable if software organizations want to adopt a quality-first approach and is key to achieving the promised results of new tools and methods.
- **Quantifiable value** is needed to keep leadership support during the investment stages of the organization's transformation. The techniques in this paper should help with communicating this in order to keep the quality-first approach on track.
- **Business partner strategic alignment** is critical to make sure the services, tools, and resources you rely on are aligned with your new long-term, quality-first goals. Specifically, you need partners that are going to adapt and react to the changes in your organization and the industry as a whole.
- **Ongoing maintenance** is often overlooked but test suites and strategies must be maintained for efficiency, and tools and services that improve and reduce the cost of maintenance should be prioritized.

SUMMARY

Calculating and communicating the business value of test automation is a critical part of enabling the leadership support and cultural change needed to adopt a quality-first software development approach. However, there is more to this transition, and organizations should be prepared for a multi-year journey with incremental improvements.

Although value is immediately recognizable in the early stages, the payoff comes from more mature adoption of tools and techniques over time. The investment goes beyond the cost of tools and encompasses training, new roles, and new strategies. These costs soon turn into gains when downtime, defects, rework, and customer support issues decrease.

Software organizations need to adopt a scalable test strategy while investing in their team, tools, and processes. As maturity and familiarity increases, so does the desire to measure success and the return on investment for their efforts. Understanding just how much money is saved is important to keep organizational momentum during the investment stages. As our customer, Caesars Entertainment, discovered, the return on investment significantly grows year over year and the value they received goes beyond tools to a transformation of the development organization.

Parasoft helps software teams recognize the value of improved software quality through a comprehensive, integrated suite of tools that reduce the overhead of manual testing and accelerate and improve the creation, maintenance, and execution of automated tests. With comprehensive reporting and analytics, all the essential pieces are in place to make the move to test automation achieve the desired business value.

TAKE THE NEXT STEP

Learn more about the results Caesars Entertainment achieved after successfully implementing test automation practices to define and measure ROI. Read the case study.

ABOUT PARASOFT

Parasoft helps organizations continuously deliver quality software with its market-proven, integrated suite of automated software testing tools. Supporting the embedded, enterprise, and IoT markets, Parasoft's technologies reduce the time, effort, and cost of delivering secure, reliable, and compliant software by integrating everything from deep code analysis and unit testing to web UI and API testing, plus service virtualization and complete code coverage, into the delivery pipeline. Bringing all this together, Parasoft's award winning reporting and analytics dashboard delivers a centralized view of quality enabling organizations to deliver with confidence and succeed in today's most strategic ecosystems and development initiatives — security, safety-critical, Agile, DevOps, and continuous testing.