VIEW POINT

MYTH VS. REALITY: HOW QUALITY ENGINEERING FUELS FASTER, SAFER SOFTWARE DEPLOYMENTS

Infosys®
Navigate your next

Quality engineering: the insurance every business needs

In today's hyper-connected digital landscape, poor software quality can be catastrophic for businesses. Imagine a scenario where a critical system failure halts operations, causing financial losses to skyrocket while teams scramble to recover. Unfortunately, this isn't just hypothetical.

A 2022 report by the Consortium for Information & Software Quality (CISQ) revealed that the cost of poor software quality in the U.S. alone had surpassed $2.41 trillion.[1] To put it in perspective, large organizations lose an average of $9,000 per minute during downtime, with sectors like finance and healthcare suffering losses up to a staggering $5 million per hour, excluding potential fines or penalties.[2]

Real-world examples paint an even grimmer picture. A CrowdStrike update once caused an IT outage paralyzing millions of Windows systems and costing U.S. Fortune 500 companies an estimated $5.4 billion.[3] Similarly, on September 8, contactless payment provider Square experienced an 18.5-hour disruption due to backend connectivity issues, crippling businesses worldwide from processing transactions.[4]

These are not isolated incidents. Software failures arise from a myriad of factors, including coding errors, ambiguous requirements, complex legacy systems, security vulnerabilities, inadequate maintenance, a shortage of resources and third-party dependencies.

Yet, amid these challenges, one key element acts as insurance against disaster: Quality Engineering (QE). QE is not just a process—it's a business-critical enabler, identifying and eliminating defects before they evolve into costly, business-threatening failures. Simply put, it is an indispensable safeguard.

While QE is crucial, it is often criticized for delaying the software development process. The typical QE phase can last five to six weeks and involves various types of testing to ensure the software's safety, security and quality. For example, executing just system integration testing (SIT) can take over 40 days.

However, this paper argues that the delay isn't inherently caused by QE but by multiple factors that organizations must address. Organizations can significantly reduce testing time while maintaining high-quality standards by understanding these factors and implementing effective strategies.

[1]. cpsq-report-nov-22-1.pdf (synopsys.com)

[2]. The True Cost Of Downtime (And How To Avoid It) (forbes.com)

[3]. CrowdStrike outage explained: What caused it and what's next (techtarget.com)

[4]. The Top Internet Outages of 2023: Analyses and Takeaways (thousandeyes.com)

Separating the myth from the reality in QE

Myth in the software industry: QE extends the software lifecycle—even in Agile environments designed to expedite delivery. The reality is quite different. **QE doesn't inherently cause delays. Instead, various dependencies and system bottlenecks can protract the QE phase.** Ideally, when development flows seamlessly, QE progresses as planned without causing delays. But that's rarely the case.

Let's delve into some of the factors that contribute to the perceived delays in QE:

The bottleneck effect:

Late-stage defect identification causes teams to backtrack, delaying releases. QE, being the final gatekeeper, is frequently seen as the bottleneck. However, the real cause of delays typically stems from upstream issues, such as design flaws or incomplete requirements.

Insufficient investment in QE environments:

In many cases, QE teams are forced to work with inadequate testing environments that don't accurately mirror production systems. While complex provisioning and maintaining these environments is critical for effective testing. When environments aren't available on time or lack the proper setup, QE faces delays often misattributed to the testing process itself.

Data issues:

Complex software systems often require intricate test data that is difficult to generate and synchronize. Manual data generation or dependencies on batch jobs from legacy systems can significantly slow testing, making it challenging to deliver on schedule.

Manual testing and limited automation:

A lack of investment in test automation results in more manual testing, which is time-consuming and labor-intensive. End-to-end automation is crucial for running tests faster and more efficiently, yet many organizations still rely heavily on manual processes.

Complex software systems:

Comprehensive testing that covers functional, regression, integration, performance and security aspects is essential but time-intensive. As the complexity of software increases, so does the testing workload.

Integration issues:

Testing becomes more complicated when compatibility problems exist between different software components. Delays in integrating these components necessitate additional rounds of testing and debugging, contributing to bottlenecks.

Inadequate early involvement:

When QE is not integrated early in the development cycle, the "end-of-pipeline" involvement leads to a flood of defects discovered in later stages, resulting in rework, which slows down the overall process.

Insufficient collaboration between developers and testers:

A lack of seamless communication between development and QE teams can lead to misunderstandings, inefficient handovers and delays in addressing defects. When collaboration falters, it's easy to see how timelines extend beyond what was initially expected.

An analysis of these issues shows that the solution goes far beyond the QE team in reducing QE cycle time. It must start with a shift in mindset at the enterprise level. This means reexamining the QE process, reassigning responsibilities across teams, introducing technical enablers like automation, and—most importantly—fostering a culture of collaboration. Closing the gap between developers and testers, ensuring early involvement of QE and committing investment in environments and tools are critical steps toward optimizing the process. With proper education, training and a commitment to change, businesses can streamline QE and speed up their software delivery without compromising quality.

QE in action – Infosys shows how

Infosys has developed an end-to-end quality engineering methodology that adds value to the software lifecycle while reducing the time required. This approach has been successfully implemented in various projects, including the following examples:

Streamlining long test cycles for a bank

A bank faced significant challenges with lengthy test cycles, particularly during the SIT and End-to-End (E2E) testing phases. The test cycles involved:

Challenges

- Implementation and system testing: 2 weeks.
- SIT and E2E testing: 24–43 days. This phase included pre-execution, build, deployment, and various SIT cycles (zero to three), each adding to the timeline.

The test cycle was lengthy due to code reviews, shakeout testing, integration issues and stability problems.

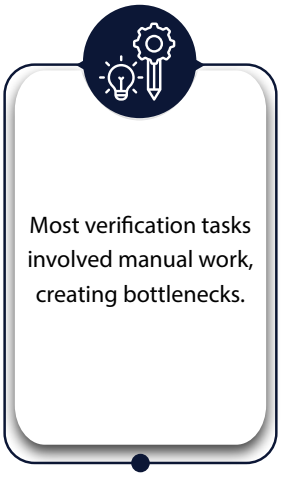
Solution

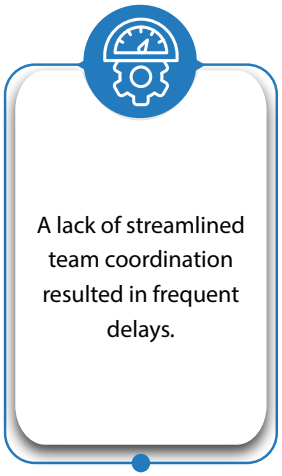
Recognizing these were symptoms of deeper inefficiencies, Infosys conducted a time-motion study to analyze how time was allocated across the testing process. By pinpointing manual tasks, automation bottlenecks, dependencies and wait times, we identified ways to reduce testing time.

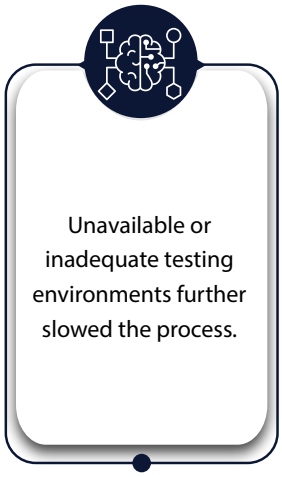
Infosys developed a solution to decrease the time spent on each testing activity. By carefully analyzing the nature of the work involved, we identified opportunities for automation, AI and other techniques to streamline the process. This significantly reduced the overall testing time by almost 60%, allowing the bank to accelerate its software development lifecycle.

Shorter E2E testing cycle for a telecom major

At a prominent telecom company, the end-to-end (E2E) testing cycle stretched to three to four months, delaying critical releases. To identify the root causes, Infosys performed a time-motion study across every testing stage, uncovering several key issues contributing to the lengthy process:

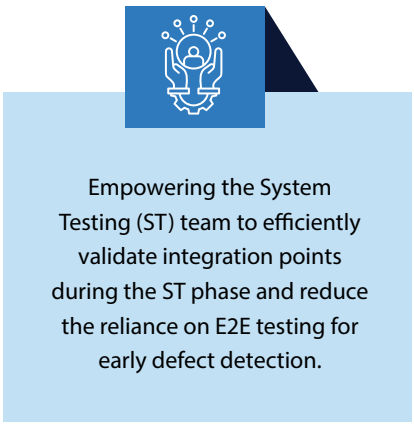


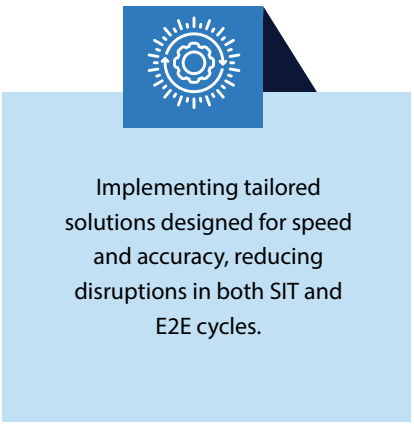




Infosys' investigation found that almost 80% of the verification process was manual, resulting in extended waiting times and coordination gaps. The manual verification process alone required eight days to complete. The actual test execution made up a smaller portion of the overall timeline.

Infosys' solution tackled these issues head-on by:





These measures significantly reduced the E2E testing cycle, allowing the telecom major to accelerate software development and time to market while maintaining high quality standards.

Emerging trends in QE

Ephermal Test Environments:

On-demand environments that represent production-like setup.

Chaos Engineering:

Ensure applications meet performance standards under various game day conditions.

Service Virtualization:

Reduce dependency on the availability of interfacing applications for completion of testing.

DORA Metrics:

AI for QE helps the customer move towards parameters that matter for business.

AI-driven QE Life Cycle:

Leverage AI for test case generation, test data creation and defect prediction to allow QE engineers to focus on value-added tasks.

QE for AI-infused Applications:

We will increasingly see QE playing a crucial role in ensuring that AI applications align with AI goals.

These trends are reshaping how organizations approach QE, enabling them to deliver higher quality software more efficiently.

Conclusion

QE is the ultimate insurance policy in a world where businesses rely on software to drive operations and innovation. The level of insurance required depends on the manual interventions during the software life cycle. QE minimizes risks, prevents costly software failures, and ensures business continuity. The focus must be on the time taken for QE, and the solution lies beyond just the QE team.

Organizations must adopt modern practices like automation, AI-driven testing and shift-left methodologies while fostering collaboration across teams to accelerate delivery without sacrificing quality. Infosys' approach shows that by addressing inefficiencies beyond the QE phase, businesses can cut costs, reduce risks, and stay competitive in the digital landscape. Ultimately, QE is not just about preventing failures—but enabling business success.

About the Author

Venkatesh Iyengar

With a career spanning across 2 decades in the quality engineering sector, Venkatesh possesses first-hand experience navigating the ever-evolving technology landscape and its profound effect on enterprises. He enjoys continuous learning, simplifying messages for stakeholders and exploring new technology and business ideas. He also holds a patent in QE maturity model design. Outside of his professional life, he enjoys long hikes, long distance runs and tennis.

For more information, contact askus@infosys.com

Infosys®
Navigate your next