Tricentis

How to modernize your application landscape and improve quality with end-to-end testing

TAKING A GAMBIT ON TESTING

Maybe you've seen The Queen's Gambit miniseries about a chess prodigy's quest to become the greatest chess player in the world. In chess, a gambit is a strategic opening move in which a player risks a pawn or a minor piece to gain an advantage in position.

It's a similar situation for software testing – sometimes minor risks are taken at the beginning of the development lifecycle to move the process along. Speed is often prioritized, and risk is pushed to the sidelines. But by leveraging end-to-end testing, companies can reduce risk from start to finish, and not make a treacherous maneuver at the beginning of their development life cycle.

However, while many organizations know that end-to-end testing is critical to a modern application environment, many are facing a checkmate in their move towards modernization.

Read this white paper to learn why you should modernize your application landscape through end-to-end testing and discover solutions to common testing challenges.

WHY END-TO-END TESTING IS CRITICAL TO MODERNIZING YOUR APPLICATION LANDSCAPE

Over the last ten years, the development and testing landscape has evolved from largely siloed processes to faster, more flexible, and more collaborative methods of software delivery. Methodologies like Agile and DevOps have allowed software innovation to soar.

Today's faster release cycles demand organizations have a testing strategy that's up to snuff. If not, you can't deliver on stakeholder demands and exceed customers' expectations. Here are four reasons why end-to-end testing is a critical component for modern digital enterprises:

1. **Application landscapes are vast and complex, and no part of them works in isolation.** To truly test whether your app and its updates can perform as intended, you must test them in the context of your entire digital ecosystem. According to MuleSoft's 2022 Connectivity Benchmark Report, the average organization uses more than 900 applications, and a single business workflow might interact with dozens of these applications via microservices and APIs.

2. **Other apps in your application's digital ecosystem are adopting aggressive release cycles**, and in a cloud-based infrastructure, your ability to incorporate those releases at your own speed has diminished. With applications releasing updates on weekly and monthly bases, the ground is constantly shifting under your feet, and your app's performance depends on the ability to test an entire business or user workflow from end to end.

3. **Legacy tools weren't built for such expansive landscapes or for iterative Agile processes.** However, you won't solve the problem by just purchasing the latest and most up-to-date tools, though better tools are part of the equation. Instead, today's environments require a modernized approach to infrastructure, development, and testing that adapts to the evolving landscape and legacy dependencies.

4. **The considerable risk if you don't modernize your approach is downtime, which can cost you and hurt your organization's reputation.** Downtime costs an average of $9,000 per minute. Some of the biggest companies in the world have experienced massive outages that caused significant financial consequences.

Industry examples

2015	A 12-hour **Apple** store outage cost the company **$25 million**.
2016	A five-hour power outage in an operations center cost **Delta airlines $150 million** and 2,000 canceled flights.
2019	A 14-hour website outage cost **Facebook $90 million**.

If companies as large as Apple, Delta, and Facebook struggle with outages and downtime, no one is safe from the consequences; developing a modernized approach that prioritizes end-to-end testing is more important than ever.

HOW TO SOLVE COMMON END-TO-END CHALLENGES

We've detailed multiple reasons why you need to modernize your application landscape by implementing end-to-end testing, but you still might be unsure about how to go about it. Many of our customers struggle with the next steps – but success can be simple. It's all about understanding the challenges of end-to-end testing, identifying the best ways to overcome them, and introducing the right processes and technology to help put those plans into action.

With that in mind, here's a look at a few of the top end-to-end testing obstacles, and how to address them.

Testing and systems complexity

The problem: End-to-end testing is so complex that even Google recommends not doing it. Their argument – end-to-end testing doesn't add value because it isn't fast, isn't reliable, and doesn't isolate the failure.

Instead, Google suggests relying on unit, integrations, and systems testing. But while integration and systems tests can help single out failures, there are likely to be bugs in an end-to-end scenario that aren't there at the systems or integration level – even if you test every integration along the path of the transaction.

End-to-end testing shines over integration tests because it mirrors a real user's actions. Without end-to-end testing, it's difficult for your team to really know if a release is ready or not. And because manual, script-based tools are so technical and outdated and systems landscapes are growing in both size and complexity, it takes your most highly skilled team members to perform end-to-end testing.

The solution: Expand automation with a codeless solution that works across both custom and packaged applications. With a codeless solution, a tester of any skill level can create automated, end-to-end test cases based on test cases that are already in their library. Consider the example of ordering a product on your website – a process that begins in a user-facing web interface but must interact with back-end inventory, order processing, and payment systems to be successful. You already have test cases that independently verify processes like user registration, adding an address, ordering a product, and verifying payment. With a codeless, end-to-end testing solution, a business tester can create an end-to-end scenario that combines these four test cases to verify the complete order process across all systems involved in the process – without ever writing a line of code.

Heavy maintenance burden

The problem: Apps are frequently added and updated within the system landscape. An increasing number of applications and more frequent updates can bury your team under a mountain of manual tests, or if you're using scripted automation, create a heavy maintenance burden. Since maintaining test scripts is just as technical as coding them, your most technical – and expensive – resources are relegated to doing maintenance work instead of creating new automation.

In some instances, your test engineers may find it easier just to create a new test rather than update an old script. But this can cause bloat and redundancy in your system, as high as 67% in some enterprise test portfolios. Over time, you'll have a hard time identifying which tests are most effective and necessary, so you might just opt to run the entire suite. This ultimately wastes time, not only by extending execution times, but also by creating false positives that must be investigated before the release is cleared.

The solution: Move to model-based to reduce maintenance. No matter how much you cull your test portfolio and reduce test scope, updating test scripts one-by-one every time the application changes simply isn't sustainable. With a model-based testing tool, you can make changes to the affected model, and that change will be automatically synchronized across all affected test cases, saving your team hours of maintenance effort and preventing bloat in your test case library.

Test flakiness

The problem: The test produces different results each time it runs, despite no change to the software code. Test flakiness often results from external conditions. When it comes to end-to-end testing, factors outside your control can cause your test to fail, such as network connections, API failures, system load, and dependencies on external systems. Flaky tests can be quite costly since they often require engineers to retrigger entire builds on CI. Worse, they can erode confidence in your tests.

You may need to pause testing to identify the source of the discrepancies. While it's paused, your team can investigate the flaky tests, identify the root cause, make the necessary updates, and run them again. Not only can this be quite costly, since it will often require engineers to retrigger entire builds on CI – it can significantly erode confidence in your tests and in your ability to assess release readiness.

Test flakiness is also affected by the testing tool itself. For example, Selenium is a useful tool, but it can create brittle tests (due to factors like data, context, and ties to external services), which makes Selenium useful, but only if your organization has the resources to maintain and update the test scripts.

The solution: Build 'digital twins' of your dependent systems. Your end-to-end tests are likely dependent on systems that are out of your reach, or outside of your organization's IT environment. Create digital twins with service virtualization to eliminate this dependency. To increase effectiveness even further, pair this approach with a model-based test automation tool, and you can ensure all your tests automatically get updated when there are changes in the virtualized service. If it's your testing tool that's causing flakiness, see the section above on reducing maintenance.

Release roadblocks

The problem: End-to-end testing is time-consuming. All the challenges we've covered are attributed to protracted end-to-end testing cycles. As the systems landscape grows and the testing burden swells, end-to-end testing will only become more laborious. For Agile and DevOps teams, end-to-end testing poses a roadblock to iterative development and feedback.

The solution: Adopt a risk-based approach. In other words, do less! End-to-end testing should be reserved for your most business-critical processes. Did you make an update to the front-end order process on your product's website? Better run and end-to-end test. If it catches an error that would've prevented payment completion or order fulfillment, it is more than worth the time and effort to prevent that lost revenue. Tricentis has found that 80% of risk can be covered by 20% of tests, so identify what the biggest risks are and how changes to your applications could introduce them, and focus your end-to-end testing there.

Access to proper test data

The problem: Your team spends too much time provisioning test data. According to Tricentis research, testers spend as much as 15 hours a week obtaining test data in the run-up to a major release. Test data comes from two places — it's either pulled from the system or made up by the testers, and both come with their own problems.

It may seem ideal to cherry-pick authentic data from your systems, but doing so adds the step of masking it to protect privacy and comply with regulations. But creating test data from scratch can take longer, and it may not produce the most realistic results or account for all the necessary factors. Incomplete, incorrect, or outdated data may cause false positives that lead to interruptions, outages, and downtime. To add to the inefficiency, siloed teams are likely duplicating efforts unnecessarily by retrieving or creating the same data as other teams.

The solution: A test data management solution that offers a centralized repository for test assets and data, eliminating hours of redundant work for your teams. Even better: A solution that can use AI to create test data as needed by observing the necessary parameters and generating synthetic data automatically, and that can easily be replicated for future testing. Ensure the solution you select is compatible with a wide variety of database technologies, file formats, message queues, etc.

Ensuring alignment across teams

The problem: Lack of alignment across teams undercuts the potential value of end-to-end testing. Because end-to-end testing spans many systems, tools, and processes, it takes a team of business analysts, developers, and testers working together to make sure it runs smoothly.

If these teams are not working in sync, it can result in disjointed communication that causes an incremental deviation from the build's purpose and goal. This issue is only exacerbated when teams use different tools that are not compatible. To avoid these misalignments, your team needs a single source of truth to facilitate collaboration and smooth handoffs.

The solution: Shift testing left and break down silos. Your technology – and your mindset – need to be updated to modernize testing. Approaches like risk-based testing and behavior-driven development (BDD) bring testing into the development and business strategy conversation. By understanding the business need and creating tests with them in mind, developers can write code that passes the test and delivers on expectations. Regardless of the methodology you choose, ensure you are clear on business requirements up front.

Remember that tools are tactical instruments, not strategic ones. Adopting a new platform won't evolve long-held ways of thinking and working. However, the tool you select should make it easier to put your approach into action as well as break down silos to foster alignment between requirements, development, and testing.

Benefits:

- Ensure transparency and alignment vertically and horizontally to minimize risk and deliver value.
- Test sooner, test faster, and test smarter with iterative Agile processes, faster feedback loops, and tools that let you scale automation and innovation.
- Enable your team to conduct end-to-end tests as thoroughly and frequently as needed.

Make a move for end-to-end testing

The Queen may be the most important piece on the chessboard because of its power to move any number of squares and in any direction. By enabling end-to-end testing, you too can have the power of the Queen and start moving your organization in a winning, modernized direction by enabling end-to-end testing.

Modernization means something different than it once did. Before, it meant volume: more data, automation, and tests. Now, we find ourselves buried in data, struggling to expand automation, and hustling to maintain test scripts. Modernization allows your team to collaborate, strategize, and innovate at their highest capacity. Enterprises that focus on revolutionizing their testing can improve quality and stay competitive.

Now it's time to make your move on the testing chessboard.

ABOUT TRICENTIS

Tricentis is a global leader in enterprise continuous testing. The Tricentis AI-based, continuous testing portfolio of products provide a new and fundamentally different way to perform software testing. An approach that's totally automated, fully codeless, and intelligently driven by AI. It addresses both agile development and complex enterprise apps, enabling enterprises to accelerate their digital transformation by dramatically increasing software release speed, reducing costs, and improving software quality. Widely credited for reinventing software testing for DevOps, cloud, and enterprise applications, Tricentis has been recognized as a leader by all major industry analysts, including Forrester, Gartner, and IDC. Tricentis has more than 2,500 customers, including the largest brands in the world, such as McKesson, Accenture, Nationwide Insurance, Allianz, Telstra, Dolby, and Vodafone.

To learn more, visit www.tricentis.com or visit one of our locations, www.tricentis.com/locations.