Tricentis

How to build a better test strategy:

A layered approach for Agile, SAFe, and DevOps environments

Introduction

The test strategy document originated in the days of waterfall development and testing – often referred to as "waterfail" because it required a huge commitment of time and people. Many organizations have abandoned this approach (and the strategy document along with it) as they transitioned to Agile and DevOps methods. Rather than doing away with test strategy documents altogether, Agile teams should revise their approach with a test strategy that is built to accommodate changes in requirements.

Documented test strategies give teams the ability to communicate and work together more effectively, regardless of the method they are using. Teams that have made the transition from waterfall to Agile may be delivering faster but may not be inherently increasing the quality of products. It has also been found that Agile can improve team communication, but less so across the entire organization. Documenting an organization-wide test strategy can address these challenges.

While the primary purpose of test strategies is to clearly define what needs to be tested and how, taking the time to define one can also spark creative ideas. In this white paper, we outline a creative approach that likens designing a test strategy to building a house. Metaphors like this can stop repetitive thinking in its tracks and become powerful catalysts for generating new ideas. Read on to learn more.

Three benefits of a unified test strategy

Communication is the number one challenge a test strategy document can address for Agile and DevOps teams. In an environment that prioritizes short release cycles, an iterative approach, and automation of repetitive steps, it's critical for all stakeholders to understand what's necessary to get the job done. Building a product according to DevOps principles means that everyone — be it a developer, operations engineer, or product manager — shares a common vision, maintaining it via communication. If the testing organization does not share their vision for quality in a test strategy, there will be gaps in communication before the project ever gets off the ground.

By following a unified test strategy, organizations can avoid these difficulties and benefit from:

1. Quickly identifying the cause when product defects are found
2. Less duplication of efforts from siloed test strategies
3. Improved efficiencies across the entire testing organization

A centralized test strategy may also result in better collaboration among teams, elimination of testing bottlenecks, and improved quality of releases.

"We don't have to approach big, complex problems like test strategy documents as if we have to boil the ocean. Much the same way that we break down complex testing problems in our everyday job, we can use these same skills to approach our test strategies."

\- Adam Satterfield, Senior Director of Test Engineering, Global Payments

Building a modern test strategy: A layered approach

The movement to Agile and DevOps has too often left test strategy documents behind. But the concept and purpose still apply, and when approached correctly, they can facilitate collaboration, speed, and overall success in modern delivery environments.

Taking a modern, layered approach to the test strategy process helps teams to understand how each testing phase contributes to the overall product quality. Each layer builds upon the last and provides cover and protection — just like with building a house. Here are the steps to developing an effective test strategy, using this metaphor as a guide.

Wearing many hats: Why communication is the strongest skill a testing leader can have

Testing leaders and their teams know more about the testing system than anyone else because they are in it on a day-to-day basis. The key to creating a truly successful strategy is to consider the different ways it may impact developer, operations engineer, or product manager roles, and to prioritize understanding their perspectives. This will help testers ask the right questions during strategy sessions and develop a shared vision for quality that everyone can endorse.

Implementing the right tools can help maintain this shared vision by simplifying cross-team communication and collaboration. Tricentis qTest improves these areas with easy-to-configure workflows that trigger updates across Agile and DevOps tools, from Jira to Jenkins. Learn more about leveraging qTest **here**.

The blueprint: The strategy session

Use the strategy session to define the test strategy, scope, and objectives for the project, ensuring alignment to business requirements. It's best to go into the session with a clear understanding of the goals: Will you use the strategy session to ensure adequate testing coverage or speed, that the final product provides an improved user experience, or that testers are clear on scope? Clarify the roles that each team member will have in testing, discuss the impacts of the test strategy on each part of the process, and decide who needs to be involved to make the project successful.

The foundation: Unit testing

A strong foundation is a prerequisite of a strong house in the same way that proper unit testing is the foundation of high quality software. Even though unit testing is something typically owned by development, the testing team must understand how it is done, where it is happening, and what metrics are being tracked. It is crucial that they know what is being unit tested and what is not.

The frame: Integration testing

Almost no part of an application works in isolation. Integration testing verifies that various units will work together when grouped. By finding defects earlier and in smaller assemblies, integration testing can reduce the cost of defects, provide structure for the rest of the testing process, and contribute to faster deployments.

Plumbing and wiring: API and database testing

Often, the biggest vulnerabilities are hidden in the API and database layers, where they can be harder to detect – so don't be tempted to skip this step. Making sure all the internal testing systems are hooked up and working can be a complicated task because it requires knowledge of both the design of the interfaces and how they are used. But with the right strategy and tools, testers can pick this up. Enabling testers to focus on the API layer will make them more effective in their roles – by catching defects earlier, where they are easier to diagnose and less risky to fix.

3 tips for working with virtual testing teams

Working with virtual teams can make strategy sessions more challenging. Here's some tips to keep teams engaged:

1. Use cloud tools to share information more easily
2. Set up micro strategy sessions with small groups instead of a multi-hour meeting with all stakeholders
3. Communicate clear deadlines for feedback so there is no need to hunt down team members

Walls, paint, cabinets, and fixtures: Automation strategy

Automation is the core of any testing strategy and the piece that everyone from leadership to testing teams are focused on. Like the finishes in a house, automation must relate to the previous layers of the test strategy and build upon them. If the goal is to find efficiencies, then the testing team must truly understand the strategy in other areas to know what efficiencies are to be found.

The finishing touches: Exploratory testing

Exploratory testing (and any other manual testing the project requires) is the finishing touch that pulls it all together. There will always be a need for manual testing: It shows testers what users will see and can help ensure the best possible user experience. Like furniture in a house, manual testing ensures users will be comfortable in all possible scenarios. Manual testing does not compete with the other layers but works hand-in-hand with them.

The neighborhood view: End-to-end testing

End-to-end testing is typically performed as part of the deployment phase. Unlike a static test that is just looking at the code, end-to-end testing builds on each previous testing layer by looking at dependencies and ensuring all integrated pieces work together as expected. To make end-to-end testing successful, bring users, business stakeholders, and product owners into the process to define high business value tests. Ask them:

- What workflows and integrations, if they failed, would bring business to a halt?
- What features or user flows, if they did not work, would cause the phones to ring off the hook?
- Which new features are most important to you, your team, or our customers?

The final test: Performance and load testing

Sometimes the components work great individually but then fail when they come together. It is essential to have performance and load testing rolled into the strategy to ensure the application offers the best possible user experience, even under in traffic conditions. This testing doesn't have to be executed every release, but should be performed regularly as part of the testing routine.

Knight Capital Group: What happens when the high business value test is skipped

Knight Capital Group was an American global financial services firm. In 2012, Knight lost $440 million in the first 30 minutes of trading. Overnight they had introduced new code that was a high-frequency trading algorithm designed to buy and sell massive amounts of stock in a short period of time. Knight called the incident a "trading glitch," but more likely it was due to problems with software development and testing models.

"*They went from a solvent, amazing company overnight to a company that was out of business by the end of the day. That is a true example of a high business value test that was not done, and the company went out of business,*" said Adam Satterfield, Senior Director of Test Engineering at Global Payments, in a recent Tricentis webinar.

Tying it all together with modern test management

With all of these different types of testing going on, often across dozens of tools, it's critical that testers have a way to track, organize, and report on testing activities across projects, teams, and tools.

A scalable, in-sprint test management solution like Tricentis qTest can go a long way towards ensuring alignment with stakeholders across the delivery pipeline. qTest's flexible, open test management can keep teams and tools connected by integrating with Agile and DevOps tools, orchestrating common events across tools like Jira, Jenkins, and GitHub via webhooks, and offering a centralized view of test automation at the team, project, or organization level.

Test management also can help put testing into the context of the business, allowing you to evaluate what's working across projects, teams, and tools. This information will become a valuable asset as you revisit and refine your test strategy over time, whether you are operating at the team or organization level.

THE MOST ASKED SOFTWARE TESTING STRATEGY QUESTIONS

1. Does test driven development (TDD) qualify as a test strategy?

It can from the standpoint of unit testing, as TDD means defining tests before writing a single line of code. In that sense, it is a fantastic albeit difficult way to approach the overall development.

However, TDD does not necessarily help when putting all the components together or understanding how performant the system is. While TDD encapsulates a great way to define unit and potentially integration tests, it often falls flat for understanding acceptance automation or manual testing. Pair it with things like behavior driven development or acceptance driven development, but it should not be the only thing in a test strategy document.

2. What are good metrics for understanding the effectiveness of a test strategy?

It is typically difficult to measure effectiveness from a document-writing standpoint, so think outside the box and consider metrics like defect data, customer health numbers, or call center numbers to capture how things are getting delivered. Capture details that help to answer this question: "Is our test strategy helping us understand our risk better and allowing us to deliver better quality to our customers?"

3. How do you combine tests without duplicating them?

Chaining tests together is often seen with performance and load testing. Instead of creating a bunch of new scripts, combine existing Selenium scripts and run them multiple times to create a load environment. In end-to-end testing, a chain of automation tests can run together to create end-to-end scenarios.

In both instances, there is naturally going to be some level of duplication. Sometimes these are expensive and time-consuming tests, and therefore not run on every release. However, defining the layers and understanding the final aim will help limit the duplication as much as possible.