

The 3 Essential Building Blocks of Continuous Testing

With a 5-Step Plan For Building a Continuous Testing Foundation

Increasing Automation to Decrease Business Risk

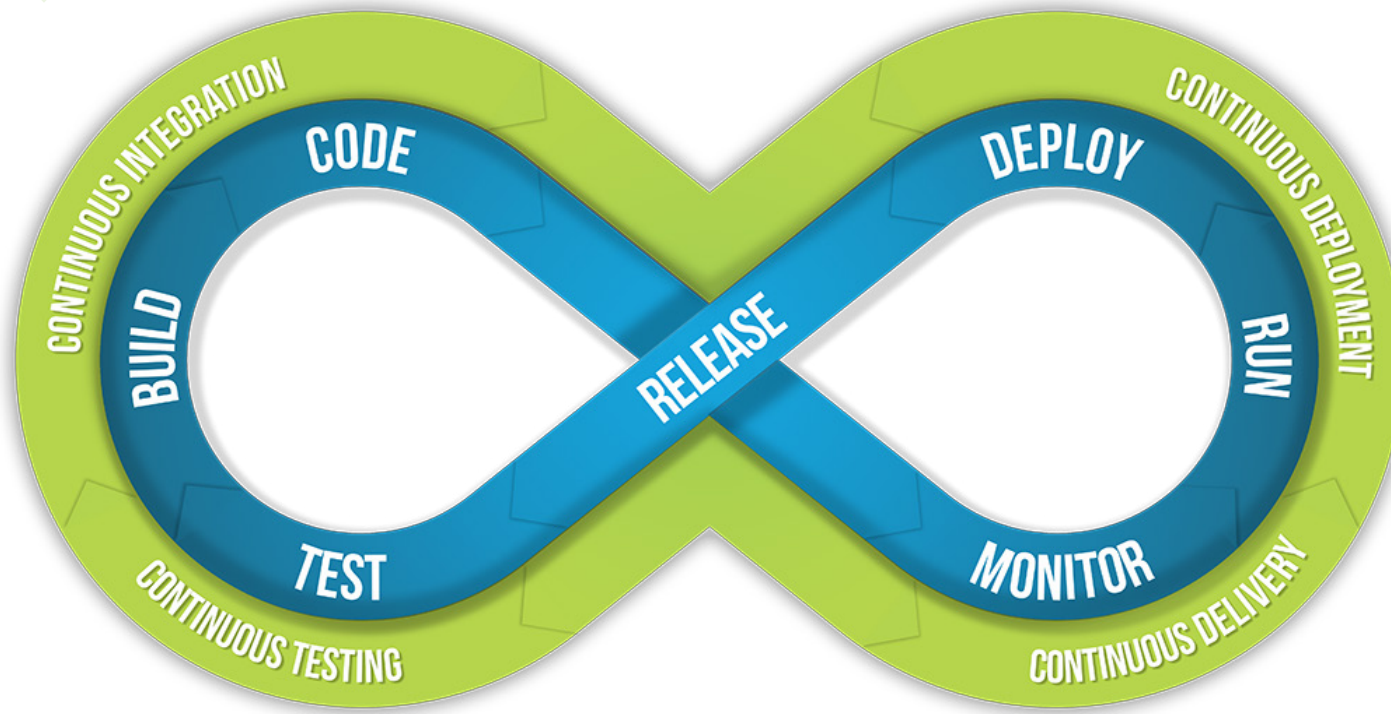
Testing is essential to ensure that your software retains an acceptable quality level and provides a great experience for your users. And automating as much of your testing strategy as possible will help improve efficiencies across your entire DevOps pipeline. At first glance, “automated testing” and “continuous testing” might seem interchangeable, however, Continuous Testing (CT), which descends from Continuous Integration (CI) and Continuous Delivery (CD) principles, isn’t primarily about automation; **it’s about business risk - especially the risk of losing your competitive advantage and loyal customer base.**

The key conceptual difference between test automation and CT is that releasing a sub-standard piece of software is bad for your business. In the best case, customers are disappointed. In the worst case, they leave and don’t come back. Continuous testing is all about making sure that despite quicker release schedules, your brand’s reputation and the user experience are protected. **Risk avoidance is at the heart of CT.**

Agile, Continuous Testing & Continuous Delivery

Although enterprises are adopting Agile methodologies for their software development, very few have actually reached a satisfactory level of test automation.¹ The philosophy of the Manifesto for Agile Software Development is closely tied to the concept of CI, which suggests that code changes should be merged often, and to CD, which dictates that software should be production-ready at all times. This is where continuous testing – and risk – step in. When more changes are made more quickly, bugs can creep in. CT is the methodology – and the mindset – of testing at regular points throughout the DevOps pipeline to ensure that quality is high enough to minimize risk to the business from releasing the software. As release schedules get tighter, automation is critical to a successful CT strategy.

¹ <https://sdtimes.com/agile/report-testing-lags-agile-development-shops/>



Test automation gives pass/fail validations of code at various points in the pipeline

Continuous testing is the process of considering whether the code is safe to release

The 3 Fundamentals of CT

Organizations – and developers – need to innovate to stay competitive, and with release schedules shrinking all the time, it's more likely that a business-threatening bug might slip through to production.

Successfully implementing CT, which aims to avoid business-killing software failures, depends on continuous alignment between **people, processes, and technology** in the organization. Each of these three are essential elements to a properly functioning CT organization.

People Challenges

When making the shift towards CT in your organization, people are crucial to your success.

In the context of CT, "people" can be broken down into roles or personas within teams – executives, developers, testers, operations, and security. Since each has a unique responsibility within the delivery pipeline, it is critical that each has the proper skills for their role and that there is a high level of coordination between them. Effective communication is key to having all parties on the same page.

Leadership needs to make sure that the various individuals work as a team and that they have the right set of tools, requirements, and continuous measurements to work with. If there is a skill set issue, it must be addressed so that there are no gaps due to lack of capabilities within teams.




Process

Continuously releasing value to the customers, with high quality, mitigated risks, and efficient feedback and analytics that can be acted upon, is a challenging task for today's DevOps teams.

Time constraints often get in the way of following common practices, such as:

- Treating staging code as production code, full version control, etc.
- Including test automation within the “definition of done” per iteration
- Committing new staging test code only when it shows consistent results
- Continuously re-factoring staging test code to make sure it is still relevant, and excluding redundant tests from CI/regression suites
- Test automation practices, use of proper object identifier strategies, and page object model





Today, there is a lot of focus on percentage of test automation - as well as test automation suite size - rather than test automation code quality, stability, and efficiency and its ability to provide valuable feedback to developers.

To correct this, executives must promote change, from measurements and metrics, to identifying issues across the various teams and ensuring use of the right tools and practices, such as including automation as part of the definition of done (DoD).

Leadership should adopt a “start small, then grow” methodology as a process. Just as teams develop their code in small chunks per iteration and stabilize them, so too should testing grow and be stabilized through automation and CI. Only then can you grow the suite.





Technology

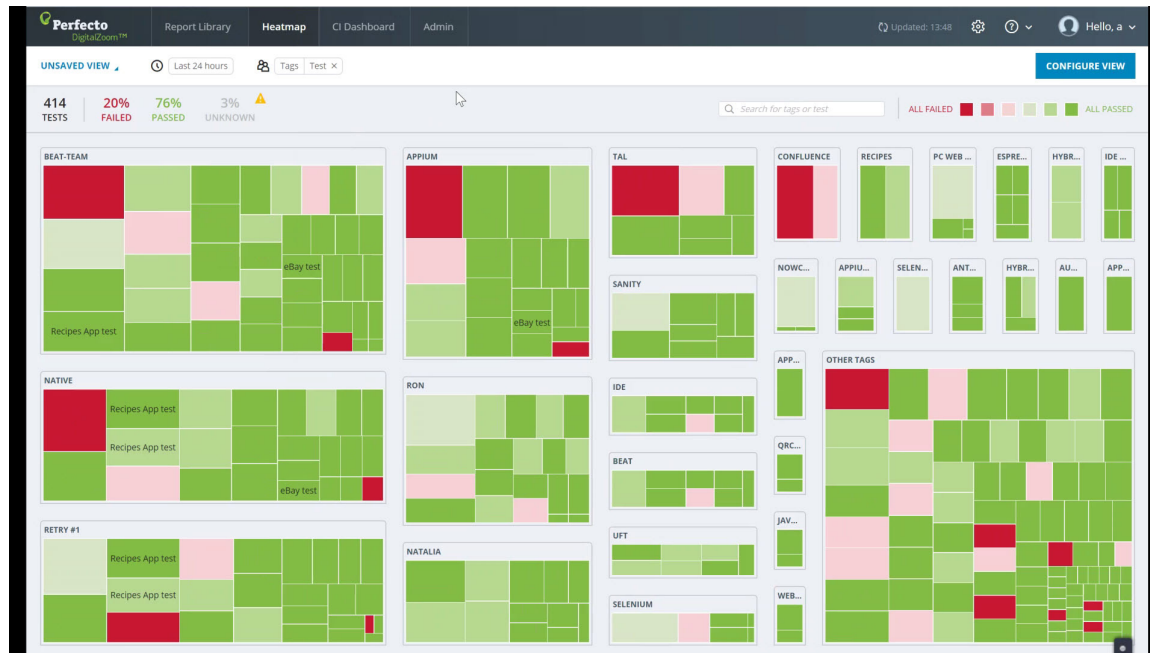
Good leadership and practices on their own aren't enough to successfully implement a CT strategy. You also need alignment with your technologies and test labs. Whenever a test environment is unstable, insecure, not scalable enough, and not easily integrated with other technologies and testing frameworks, teams can fail to meet their DevOps objectives.

In an Agile environment, where various developers and testers use different technologies and platforms, and sometimes are composed of remote teams, having a stable test environment upon which to base all quality activities is a key for success. It is not just about matching the test lab and the environment to the tools of the team, but also matching it to the teams' skills and software methodologies, such as BDD, ATDD, TDD, or others.

Test labs that cannot fit seamlessly into development CI processes cannot serve the requirements of continuous testing. The stability and reliability of CI relies on the lab as its backbone.



In the era of DevOps and CT², there is much more test data that is being generated per each execution. Gaining insight into large amounts of test data and a deeper understanding of specific issues can be a key enabler for implementing CT. Executives can't reach a "go, no-go" decision without executive dashboards;



developers and release managers struggle to see whether their CI processes and builds are moving in the right direction and remaining within time constraints. In many cases, pinpointing an issue feels like finding a needle in a haystack, and test engineers and test managers might lack the necessary test artifacts to provide as much feedback as possible to developers.

Successfully implementing a CT approach is easier said than done. There are many moving parts in the pipeline that can slow down or entirely disrupt your quality release activities.

² Cloud Computing – making CI, CD and CT work together - <https://www.cloudcomputing-news.net/news/2018/apr/11/finding-right-agile-formula-making-ci-ct-and-cd-work-together/>



8 Considerations for Executives When Planning for Continuous Testing

When planning for a continuous testing strategy, leadership should have a comprehensive plan for all teams and individuals:

1. Number of projects and their type (mobile, web, responsive web, progressive web, etc.)
2. Team size – dev, test, ops, source code management, etc.
3. Team skill sets – development languages, testing framework familiarity
4. Software development lifecycle methods – ATDD, BDD, etc
5. Technology availability – which tool stack is currently in use and what is missing?
6. Market trends and analytics in order to support new features, user stories, etc.
7. Lab sizing and coverage requirements
8. Metrics for success which are achievable, meaningful, and clearly understood

“Plan your work for today and every day, then work your plan.” ~ Margaret Thatcher



A 5-Step Plan for Building your CT Foundation

After setting goals and quality objectives for your projects, the various individuals in your team must have a solid foundation under them in order to meet these goals. This foundation relies on a few important elements:

- Test automation framework and test authoring
- Test framework fit with your teams' requirements
- Risk-based test automation strategy
- Stable CI maintained continuously
- Support for test data generation/analytics
- Stable lab and test environment






1. Test Automation Framework and Authoring

Authoring test automation code should be no different than developing and maintaining code for your application features, such as login functionality, etc.

As the product evolves, so too should your test scenarios. When the test code is not properly maintained through source code management (SCM) tools, it is hard to make timely changes and build on top of them since there is no proper documentation, traceability, or history of the test code.

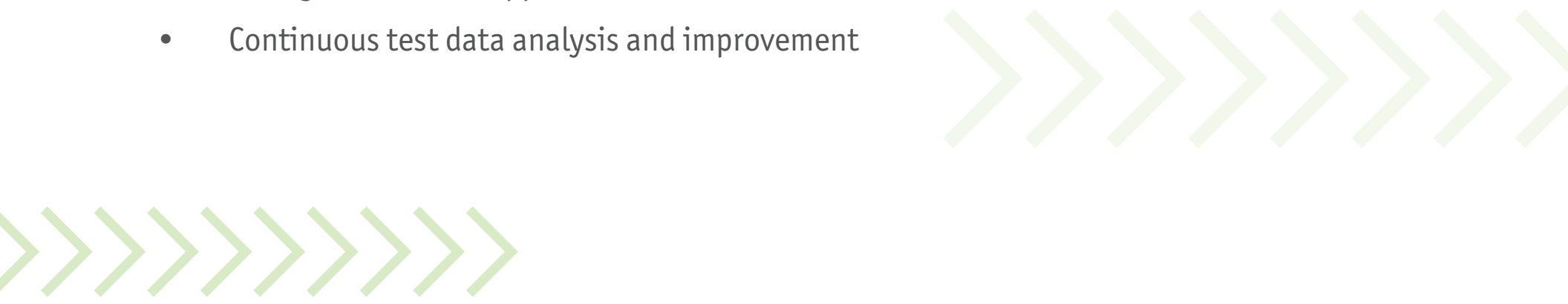
In addition, when a new functionality is built into your mobile application or your web site, this requires changes to your test code that may include writing new tests, modifying existing test flows, retiring tests, or maybe even merging new tests into one large test. To be able to continuously address product changes, your test code always needs to be in sync with the product's evolution and carry with it relevant history and documentation.





Just as you treat your production code with ongoing analysis, refactoring, and more, it is critical that you do the same for your test code. This is an enabler for CT.

Developing your test code the right way from the start can mean the difference between a high-value test suite and a time-consuming, flaky and inefficient one. As mentioned, treating test code as production code is recommended. Writing good, valuable tests is a different practice that involves various considerations, including:

- The right object identification strategy
 - The right test framework to work with
 - Measuring test efficiency within the CI
 - Taking a risk-based approach to test automation
 - Continuous test data analysis and improvement
- 

2. Risk-Based Test Automation Strategy

A proper CT strategy offers feedback and the best possible experience to end users. However, these are only part of the equation. An organization must always be considering the potential consequences of releasing software that might damage the all-important user experience. This is the real mission of CT.

One possible approach to developing a CT test automation strategy would be to embrace the best practices [Angie Jones³ laid out](#):

- What's the test engineer's gut feeling?
- Risk calculated as probability to occur and impact to customers
- Value – does the test provide new information and, if failed, how much time to fix?
- Cost efficiency to develop – how long does it take to develop and how easy is it to script?
- History of test – volume of historical failures in related areas and frequency of breaks

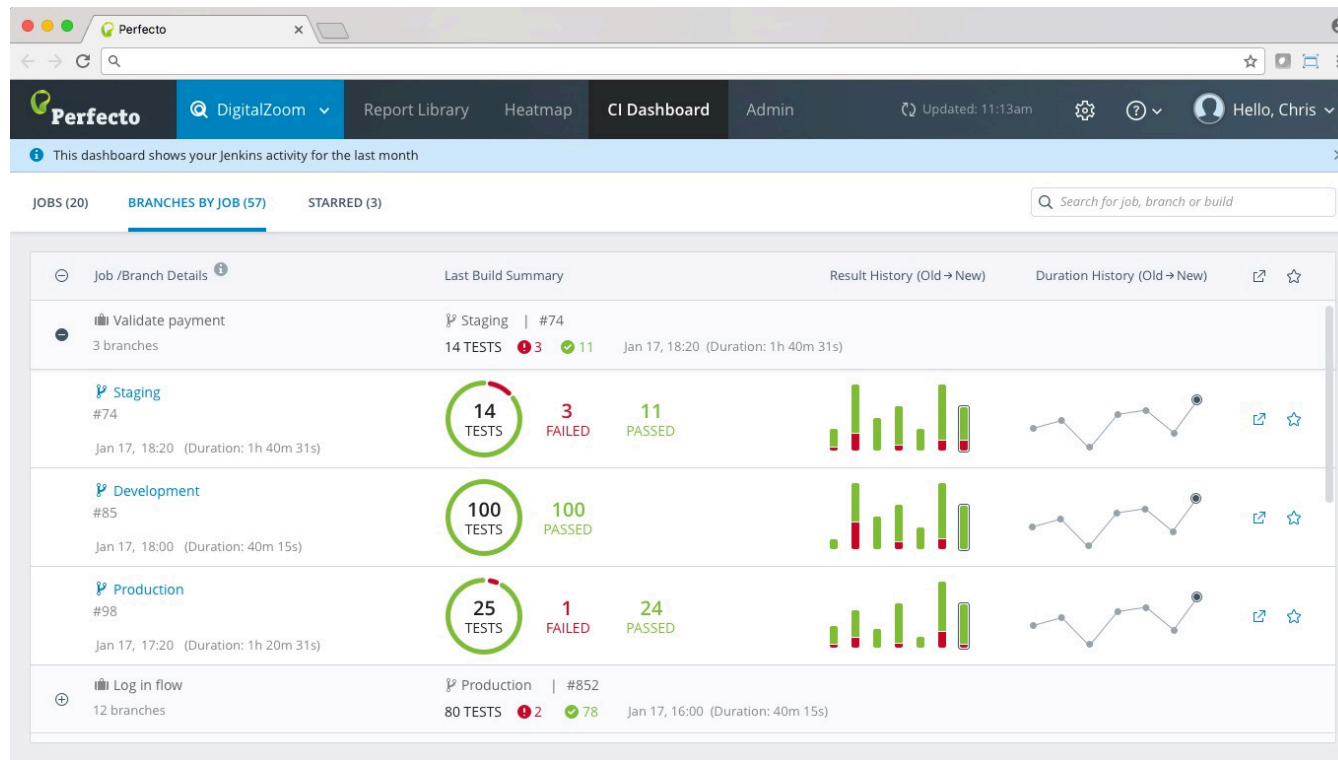
It is important to use either this or another method that makes sense from quality, business, and time-to-market perspectives to drive efficient, ongoing CT processes.

³ Angie Jones slide share on what to automate - <https://www.slideshare.net/saucelabs/which-tests-should-we-automate-by-angie-jones>

3. A Stable CI

The essential components for achieving efficient DevOps are CI, CD, and CT.

If Continuous Delivery and Continuous Testing are the wheels on a car, then CI is the engine that drives them. That's why continuous integration builds should always be green unless there is a real issue. To make sure your CI is always green, teams need to focus and develop processes that decide what gets into that engine.



4. Test Data Generation

Depending on the market segment or vertical, test data generation offers high value to both developers and testers. For banks, the ability to continuously test against pre-generated banking accounts that have various profiles is key to validating both functionality of their apps as well as performance in pulling account data. The same goes for insurance apps and healthcare apps. Such test data cannot be actual production data for a wide variety of reasons. Therefore, as part of CT, teams need to have a way to virtualize and use this test data. Also, when the product evolves and test data becomes outdated, there must be a clear process for refreshing the data to include new fields or other advanced features.

5. 24/7 Stable Lab and Test Environment

As the entire CI process relies on well-written test code, both test execution and CI stability depend on the availability and stability of the test lab. A stable lab means that all platforms connected to it, together with supporting test environments, 3rd party tools, and APIs, are working 24/7 with near-zero downtime. To create such an environment, in the DevOps⁴ landscape, there is no doubt that the cloud has proven to be the only practical way to meet these objectives.

⁴ Mabl blog – mapping the DevOps tool chain - <https://www.mabl.com/blog/devops-tools-organized-by-category>

Start Small and Grow Your CT

Incorporating CT into your DevOps pipeline means always having the right coverage at each stage, both from a test scenario and a platform perspective.

Each phase of DevOps has different objectives and each of these require different coverage principles. The early stages of developing a new functionality involve many unit test activities and back-and-forth debugging cycles by the developer. Typically, these activities happen on the developer's workstation and, to get fast feedback, the developer doesn't need a full-blown lab and will be satisfied with one or two relevant platforms. As development of the feature(s) evolves, there are different quality and development criteria which require an extended sets of test scenarios and platforms to be executed; this requires a scheduled CI workflow backed by a larger test lab (that will probably be cloud-based) to provide support for various users, various scales, and a more complex test environment.

Continuous testing within the DevOps pipeline should be built according to each different phase, from early development through production monitoring. Each phase will include its relevant testing activities which will be triggered based on the activity and quality objectives via the relevant environment (dev workstation vs. CI server) – per commit, nightly regression, and ongoing monitoring. For each, and based on the team's process, skills, and CT plan, teams will need to have the appropriate test framework for coding (Espresso, Appium, XCUITest, Protractor, etc.)

Also, CT triggers and environments do not define the testing scope; for this, each project needs a solid test plan and quality criteria that strike the correct balance between API, unit, UI, functional, and non-functional testing.

Minimize risk - Develop the CT Mindset Across the Organization

We've laid out the background, theory, and some best practices for implementing CT in your organization. Implementing CT is not an easy process. There's not a button you can push to make it happen. If your organization isn't there yet, you're not alone. According to a survey conducted at Cloud Expo Europe, 43% of organizations haven't yet adopted DevOps.⁵ Continuous testing is a mindset that needs to be fostered across your whole organization, from top to bottom.

CT is about incorporating a testing mentality at every stage of the SDLC pipeline in order to minimize risks to user experience, and at a more fundamental level, your company's reputation. Continuous Testing is not something that you do when everything else is done. It needs to be part of the process. With integrations built for nearly every testing framework, unparalleled quick-view visual analytics, and built into all major CI platforms, Perfecto has the tools to help ease your transition to a successful Continuous Testing strategy.



⁵ <https://medium.com/ecs-digital/7-devops-adoption-statistics-for-2017-infographic-ca368065a7c7>

Appendix: Glossary of Terms⁶

- **Test automation:** Designed to produce a set of pass/fail data points correlated to user stories and application requirements
- **Continuous integration:** The practice of merging all developer working copies to a shared mainline several times a day
- **Continuous delivery:** The evolution of continuous integration by always being able to put a product into production
- **Continuous deployment:** Taking continuous integration and continuous delivery for automatic deployment to production
- **Continuous development:** The umbrella term of continuous integration, continuous delivery, and continuous deployment
- **Continuous testing:** The process of executing automated tests as part of the software delivery pipeline in order to gain feedback on business risks with a software release candidate as rapidly as possible
- **DevOps:** The software engineering practice that aims at unifying software development (Dev) and software operation (Ops)

⁶ <https://www.techwell.com/techwell-insights/2018/01/whats-winter-2018-issue-better-software-magazine>

About Perfecto

Perfecto enables exceptional digital experiences. We help you transform your business and strengthen every digital interaction with a quality-first approach to creating web and native apps, through a cloud-based test environment called the **Continuous Quality Lab™**. The CQ Lab is comprised of real devices and real end-user conditions, giving you the truest test environment available.

More than 1,500 customers, including 50% of the Fortune 500 across the banking, insurance, retail, telecommunications and media industries rely on Perfecto to deliver optimal mobile app functionality and end user experiences, ensuring their brand's reputation, establishing loyal customers, and continually attracting new users. For more information about Perfecto, visit www.perfecto.io, join our community follow us on Twitter at **@PerfectoMobile**.

Start your free trial with Perfecto today!

