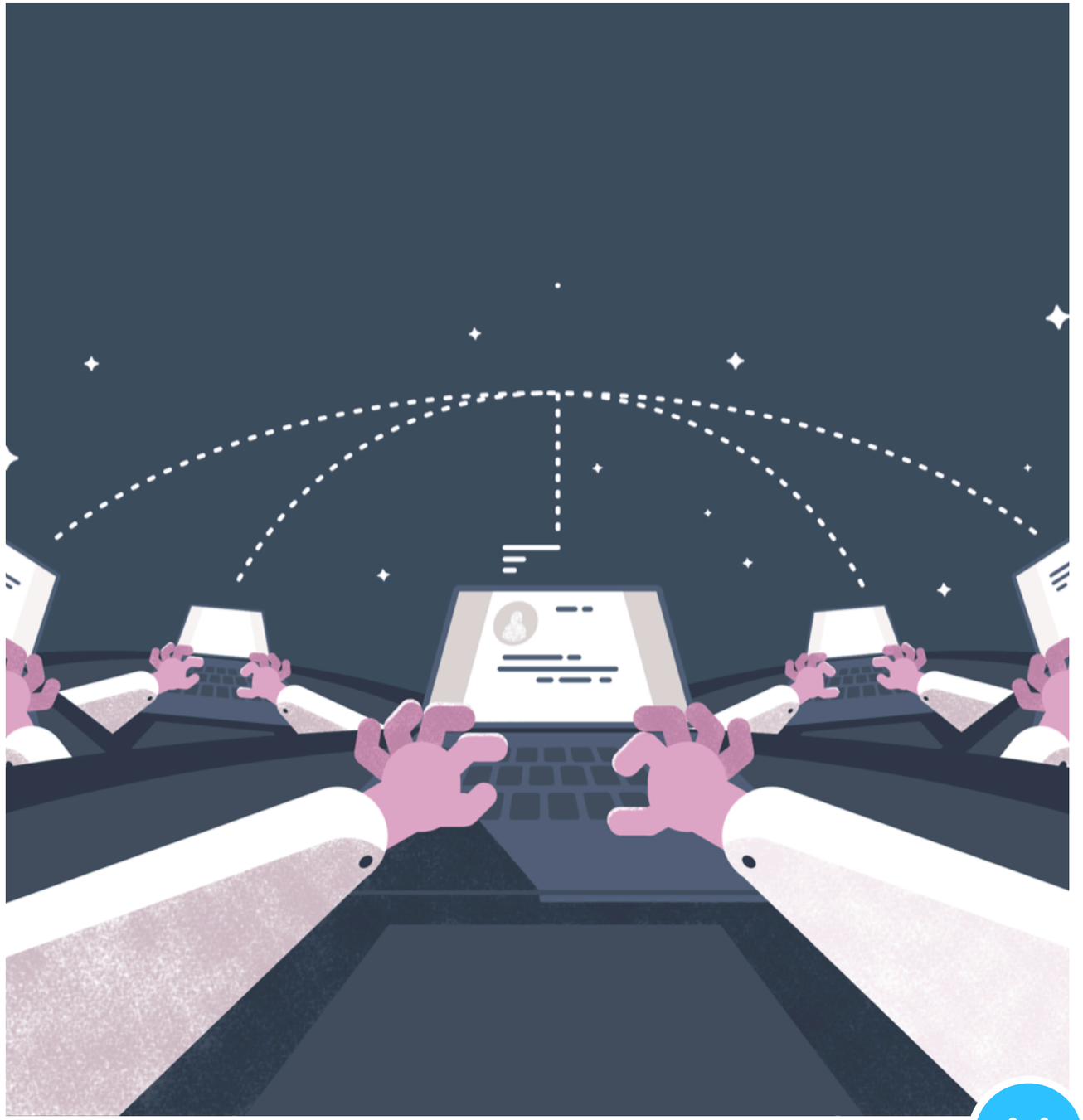
CI/CD: The Solution To High-Cost Quality

www.testim.io

Testing processes have evolved drastically over the past 40 years. As software becomes more complex, traditional software development processes are beginning to fall apart. Manual testing by individuals is forcing organizations to choose between cost, user experience and time-to-market.

Luckily, there's an answer. Continuous integration (CI) and continuous delivery (CD) are key practices in modern software development processes. CI and CD work together to ensure successful development operations. Automating build, integration and testing processes can help you optimize your development pipeline to weed out bugs early on and fix them as quickly as possible at a lower cost to your business.

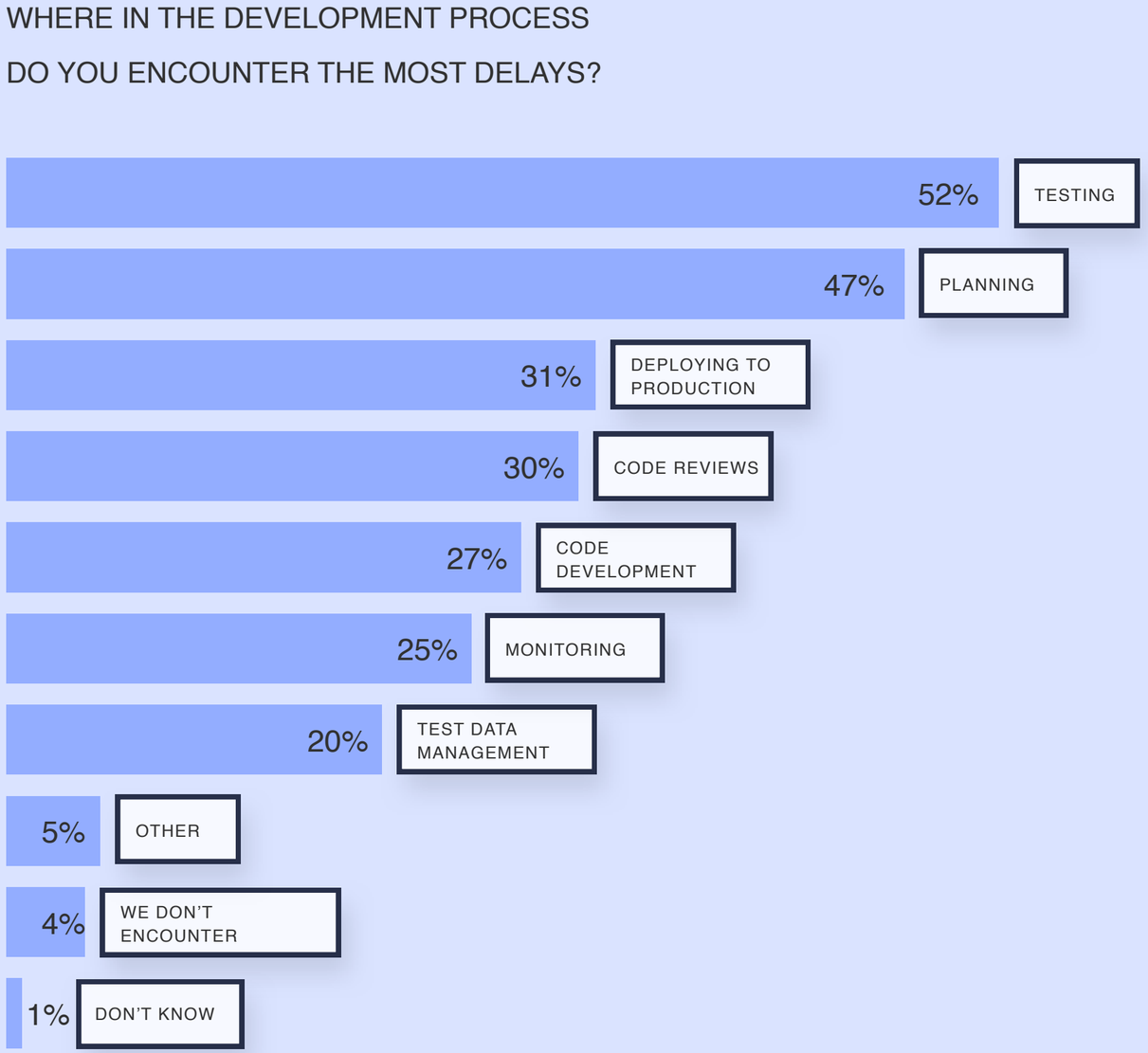
In this guide, we'll talk about the real cost of quality and the many ways in which automated testing can drive those costs down. We'll also dive into some key testing strategies, including how to decide what to test, who should be in charge of testing, and the frequency of testing.



How much does quality cost?

Any developer knows that no bug fix is free. From the cost of labor to the cost of licensing, lab, energy and deployment, ensuring the quality of an application can send thousands of dollars and hours of productivity down the drain. According to the 2018 Global Developer Report, more than half of respondents said they encounter the most delays during testing.

These delays in testing can be expensive. The closer to production bugs are found, the higher the cost of fixing them. Multiple studies by software companies, academics and large banks have revealed the staggering financial differences in discovering bugs in production and discovering them in development. According to a study performed by Bank of America:



- Discovering a bug in development is the least expensive. Fixing this bug would cost about five hours and $390. If a test fails in development, the code and its implications are still fresh in the developer's mind, allowing them to solve the issue quickly and continue pushing toward the release date.

- Discovering a bug in quality assurance (QA) is about 30 times more expensive to fix, costing up to 127 hours and $11,700. This huge jump in cost is due to the extra time and effort it takes to fix. The QA person has to report the bug to the senior manager, and the senior manager has to assign the fix to a developer. Switching context, taking the developer away from his current sprint and back to a feature he previously wrote is time consuming and can potentially impact the current sprint.

- Discovering a bug in production is estimated at $72,000. This huge expense is due to the cost of triaging, reproducing, setting up the environment all over again, developers stepping back into a code, potentially one another developer wrote. It can take QA days to understand what's causing these bugs and the developer now has to switch context back to a feature he forgot about long ago. While your team is working, your business suffers.

One way to battle these costs is by testing as close to development as possible with a fast and continuous feedback loop. Continuous Testing (CT) is key to achieving that. Continuous testing ensures fast builds, fast tests, and shorter feedback between the developer writing the code and the developer testing it.

Organizations who've deployed continuous testing have noted a decrease in the cost of quality, an increase of agility and an increase in on-time delivery. The more quickly developers are alerted of bugs, the more quickly they can fix those bugs and the lower the cost of quality.

The role of CI/CD

CI tools allow you to automate the entire process from development to production and test throughout the process. It also allows you to pull all of the changes made by different developers and put them together to create a master version of the application. Finding bugs early on requires testing each of these commits against the master branch. And in the basic software world, manually testing each of these commits on separate machines may run great.

But in the real world of software development, when you have fifty test cases you need to run daily, you need a more stable solution — you need to shift left. And test automation is key to achieving the level of continuous testing you need.

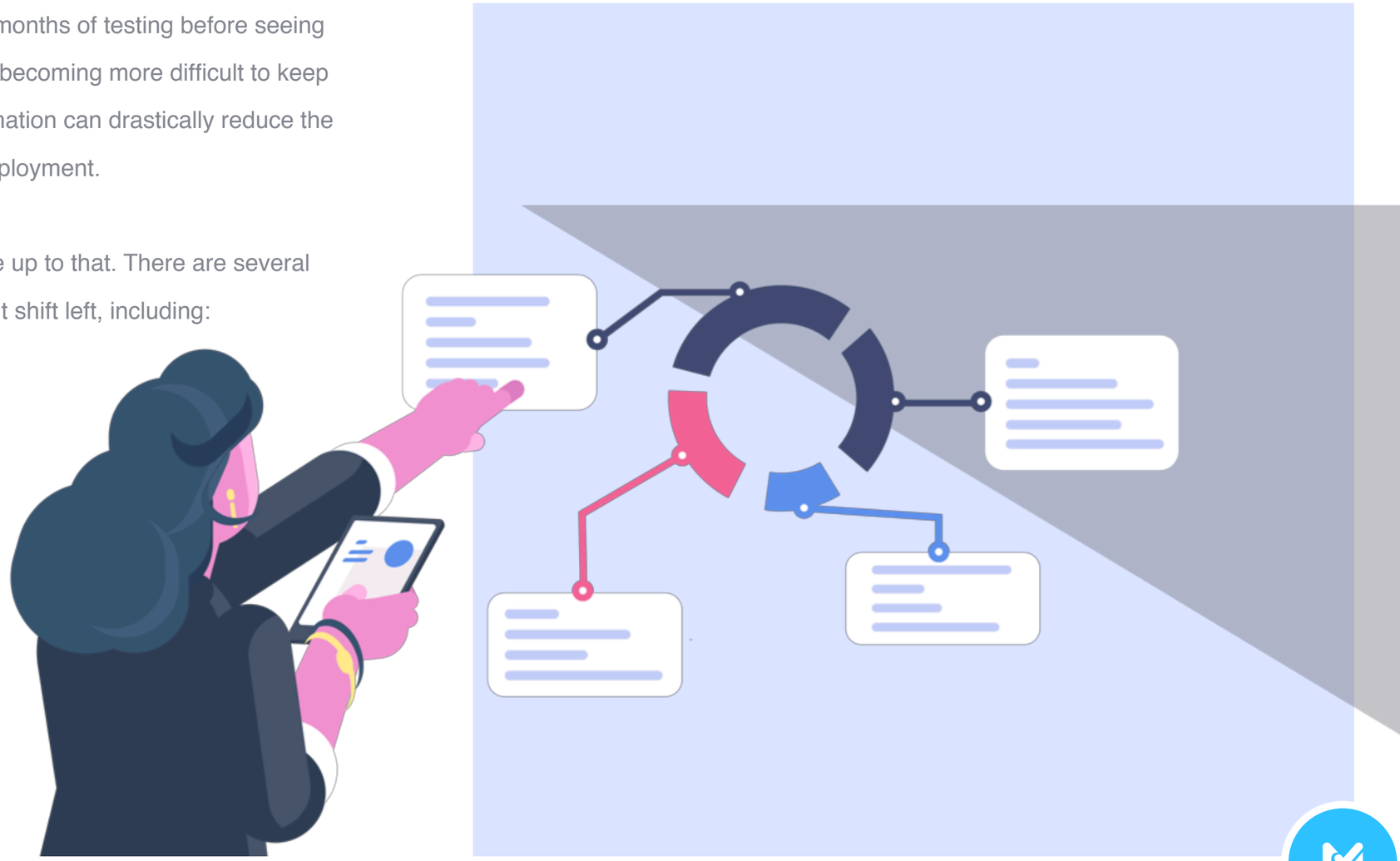
By automating the testing process with CI, you can run multiple concurrent tests across browsers. Furthermore, CI provides you with the desired capabilities remotely, rather than being confined locally to one machine.
It's more scalable, more agile and more cost-effective.

CI gives the developer the ability to test and automatically deploy their changes into the master branch without breaking the entire code. Before any change is committed and merged into the master environment, CI/CD tests for bugs as early as possible, reducing the amount of time and money it takes to fix them — and ensuring a quality application every time.

Challenges of automated testing

Traditional software development processes require months of testing before seeing any results. As software becomes more complex, it's becoming more difficult to keep up with the amount of testing required. But test automation can drastically reduce the amount of time needed to test and fix bugs before deployment.

But not a lot of organizations haven't been able to live up to that. There are several challenges you may run into when trying to implement shift left, including:



- Skillset: Generally speaking automated tests require coding. Just like coding your app, automated tests require organizing your code, setting data, importing libraries, defining classes, etc. An automation engineer is a developer by skill-set. One the one hand, transforming manual testers to become automation engineers is a tough transition and is likely to fail. It's a different persona. On the other hand automation frameworks are based on open source frameworks that your frontend developers are most likely not familiar with.

- Authoring: Authoring good tests takes time, and even skilled developers may take days to write a single test. While authoring tests, developers need to think about how to handle element weights, make reusable components and parameterize the tests.

- Initializing: Many companies hit a wall in the process of initializing data and preparing it for the tests. Automated tests depend on the integrity of their initial state, so if you can't break down this wall, you can't successfully implement automated CI testing.

- Maintenance: The stability of a test depends on the stability of the application. If you change the screen of an application, the tests start to break, so you have to change each test to accommodate those changes. Maintenance can add up to 30% of a tester's overall work, but automating tests can streamline this process.

- Scalability: Some companies run into issues when they try to scale their businesses. As the application becomes more complex and you need to release more often on a larger scale, you'll need automation.

To overcome these challenges and begin using CI/CD properly, you need to transform your company's testing mentality and development processes.

4 essentials for continuous testing

Once you've overcome the main challenges of implementing shift left, there are still certain strategies you need to employ. So how do you implement continuous testing effectively? Here are four key ingredients to streamlining your development process:

ORGANIZING YOUR TESTS

To optimize your QA processes, segment your tests into suites and run different suites at different stages of your development cycle. Full regression might include thousands of tests, but running those tests every time a developer pushes code costs an enormous amount of time and resources.

If two to three key tests failed, there's no point in waiting for the full suite to complete. Most organizations run a subset of their scenarios on every commit with a nightly full regression. Figuring out which scenarios should be included in your sanity and which are full regression is not an easy task. High value scenarios, ones that a prone for bugs and parts in the code that frequently changing are good candidates for your sanity. These should be short, concise and specific. The more extensive suites can be saved for after a successful sanity suite.

TEST AUTOMATION

It's impossible to gain the right amount of coverage with just manual testing and it's not feasible to hire hundreds of testers to speed up the process.

The best way to speed up the authoring, execution and maintenance of tests is to automate them with CI/CD. Automated tests can run 24/7 across multiple projects, providing optimal coverage for QA. Ultimately, this will increase on-time deployment and ensure quality, all at a lower cost to your business.

REMOTE RUNS/GRID

Once you have dozens of tests you cannot run them on your QA's local computer, especially if you want to test multiple browsers. A Selenium grid is used to run tests remotely and provide the results back to the team. A Selenium grid can spin up and down different combinations of browsers and operating systems. A large number of such instances is required to run tests in parallel and provide a short feedback loop. The time it takes for the dev team to get feedback is a combination of the number of tests that you run as part of your sanity, the length of those tests and the number of tests you can run in parallel. By increasing your levels of parallelism, you shorten your feedback loop — ultimately increasing on-time deployment and reducing overall development costs.

BUILD-TEST-DEPLOY AUTOMATION

The software development cycle includes dozens of handovers across different stages. Human intervention, of course, can always delay this process. By shifting left, you can automate this handover and optimize the entire process.

Automating handovers requires Continuous Integration (CI) and Continuous Delivery (CD) servers. As you know, CI allows developers to monitor events in the development cycle and triggers a set of actions once certain events have occurred. For example, CI can enable developers to merge their changes back to the main branch of an application as often as possible. Then, they can program the CI server to trigger automated tests against this new build. It checks that every new commit integrated into the main branch doesn't break the application.

While CI deals with the build-test part of each development cycle, CD focuses on what happens with a committed change after that point. CD allows developers to release these new changes quickly and sustainably. So along with the ability to automate your testing, you can also automate your release process and deploy the new application at any time. This allows you to deploy in small batches that are easy to troubleshoot in case of a problem.

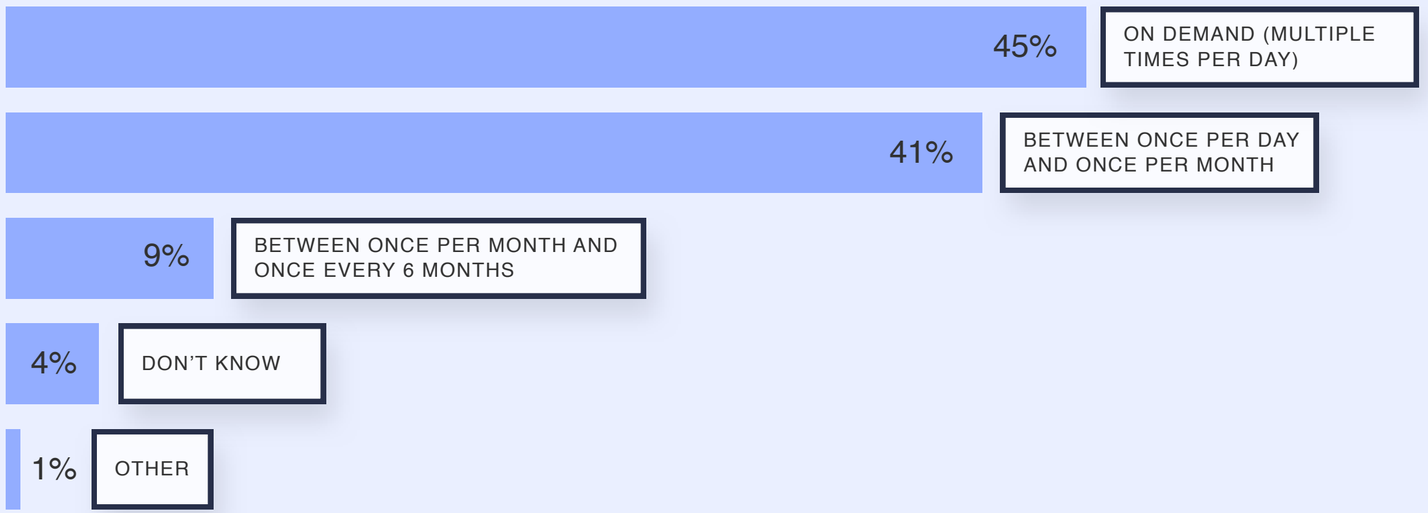
But to work in a CI/CD environment, you need extremely stable tests that you can trust to run quickly and frequently. You need to have the proper infrastructure of automation to minimize false and flaky tests.

So how often should you test, anyway?

It can take significant amount of time to author and execute tests depending on how often your test. If you're testing every six months, a single test can take a week to complete, whereas weekly tests might only take a day. In the end, manual testing takes far too much time and money, and it's not sustainable in modern software development processes.

Most organizations already know this. According to the 2018 Global Developer Report, almost half of organizations deploy code multiple times a day, requiring a greater frequency of testing as well.

HOW OFTEN DOES YOUR ORGANIZATION DEPLOY CODE?



Ideally, developers should run tests on every commit. Segmenting them into automation suites help to ensure the right set of tests are triggered each time. The really innovative companies are running between five and 100 tests a day — that's a lot of tests to juggle without automation. Using CI/CD, you can shrink your testing cycle and free up developers to focus on sprints.

Ideally, developers should run tests on every commit, but segmenting those tests into automated suites so that the right set of tests are triggered each time. Those sets shouldn't be so small that you don't have enough coverage, but not large enough that you don't have the QA and developer capacity to check them either.

Keep in mind that tests that fail require attention. Testing too frequently without allocating capacity to triage failed tests and fix bugs is useless.

Speaking of coverage...

When you've decided to commit to continuous testing, it's important to define optimal coverage. As you know, applications can have thousands of different scenarios. Some are more common than others and some are more serious. In an effort to ensure quality, some organizations may try to test every possible scenario. But this leads to hundreds of tests of lower quality and little value.

What you test should differ depending on the type of application and whatever's driving your business. Do you care more about the functionality? The way the user interface looks? You have to think about how critical certain scenarios are to the application, and then segment those scenarios to ensure you're getting the right coverage with each test.

If you're a company like Coca-Cola, for example, you definitely can't afford a bad visual experience — so maybe you'd do a UI validation down to the pixel of your logo. But if you're developing a business application, then you probably care more about the functionality of the application than the way it looks.

Regardless of what you decide, there are three areas that every organization should test:

- Critical paths
- Stable parts
- Revenue generating flows

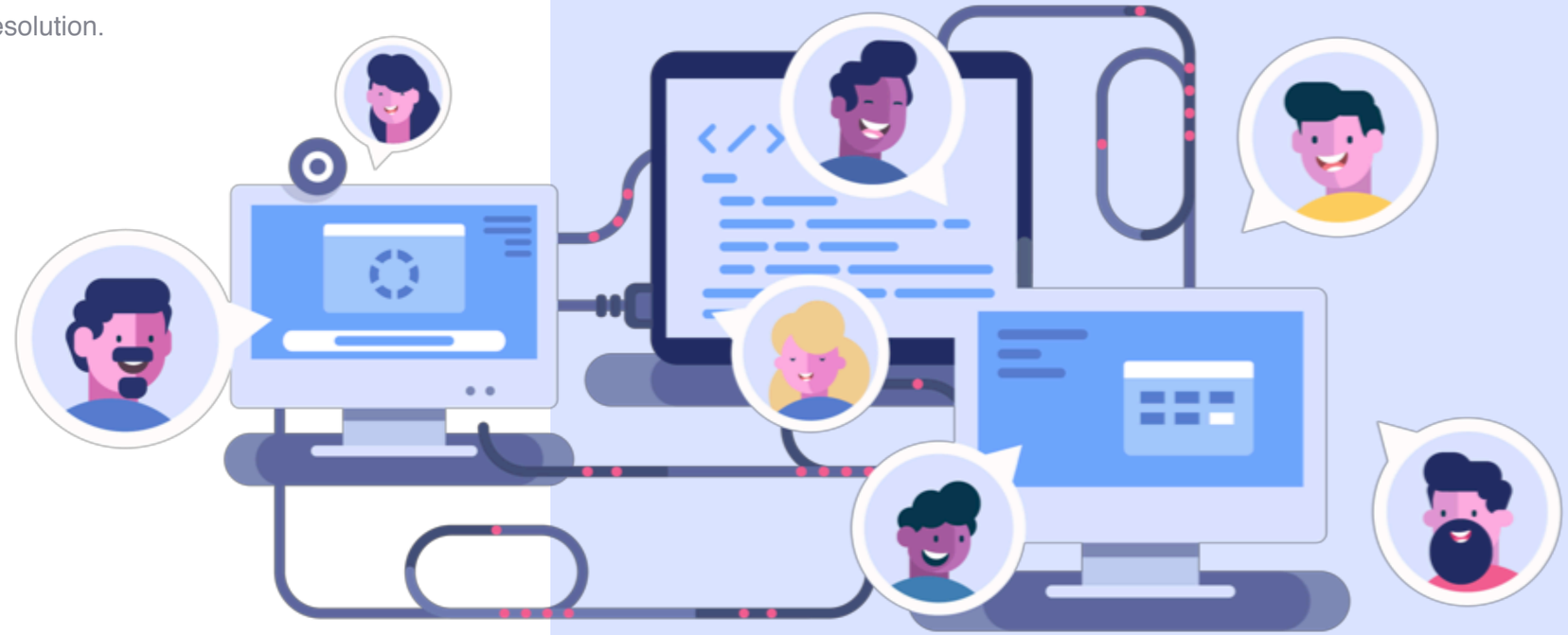
These are the essential parts of your application, the paths that current paying customers move through most frequently, such as the login. If the login fails, you'll lose business and take a hit on your revenue. In other words, this path is critical to your business. You can use analytics tools like Google Analytics to see which of these user flows are most common, and start listing out the ones that are most important to user experience and revenue.

And there's no reason that only the QA person should be running these tests. Use regression testing throughout each stage of development to make sure specific, critical functionality always works. Exploratory testing can be used on a lower frequency by manual QA to find bugs in other areas.

Remember: your tests should be short, concise and specific. With automated CI/CD, you can ensure you're squeezing maximum value, coverage and efficiency from each test.

Debugging - The overlooked aspect

Test automation is just one part of the cost of quality. The other part is the cost of fixing bugs. The real work starts when a test fails. Reducing cost and feedback loop requires rich information to quickly determine the root cause. Screenshots, log files, detailed error messages and the ability to reproduce can reduce triaging time. Integrating your tests to your bug reporting tool such as Jira, Trello or Slack can help teams communicate better, leading to shorter time to resolution.



Quality is everyone's responsibility

Many organizations still operate under the mindset that only the QA department can ensure deployment of the highest-quality product. The role of the QA engineer is a big one, and it's not an easy job, especially when QA isn't pulled in until the last stage of the DevOps process. More emergent companies, though, are acknowledging the significant role developers can play in ensuring quality. Some organizations are shifting left i.e having their developers take a significant part in testing.

Why? Because developers know what changes they are making to the code and therefore know the critical paths that need to be tested in a given timeframe. It's easier for them to make changes to the tests as they make changes to the application. By automating this process with CI/CD servers, developers and QA engineers can work together to make sure these tests are as stable as possible.

Traditionally, developers have taken a reactive approach to QA. They make changes to the code, the QAs' tests fail, and they go back to fix the bugs. But testing can no longer be a secondary issue. By taking a more proactive approach to testing with CI/CD tools, developers can smooth away many challenges of the testing process and drastically improve quality, all while reducing the overall cost of the DevOps pipeline.

Proactive means making changes to the tests as they make changes to the application. This way, functionality is sustained and the tests go smoothly. Or, organizations can take a hybrid approach to this, where QA creates the first batch of tests, and the developers each maintain their own area of the application. Implementing these processes with testing automation ensures minimal time and maximum efficiency.

Ultimately, though, everyone should have quality in mind, and everyone should be testing at each stage of development. Without a company-wide value placed on CI/CD, it will be impossible to shift left effectively.

Whoever's handling the product at any stage of the process is fully responsible for sustaining functionality during the handoff. Each stage of development should involve some sort of regression testing for quality assurance. Many of the important bugs and critical issues QA would find could be resolved in the earlier stages of development by briefing developers on where QA would test to break their codes. By working together to author automated tests, QA can be involved during every stage of the process.

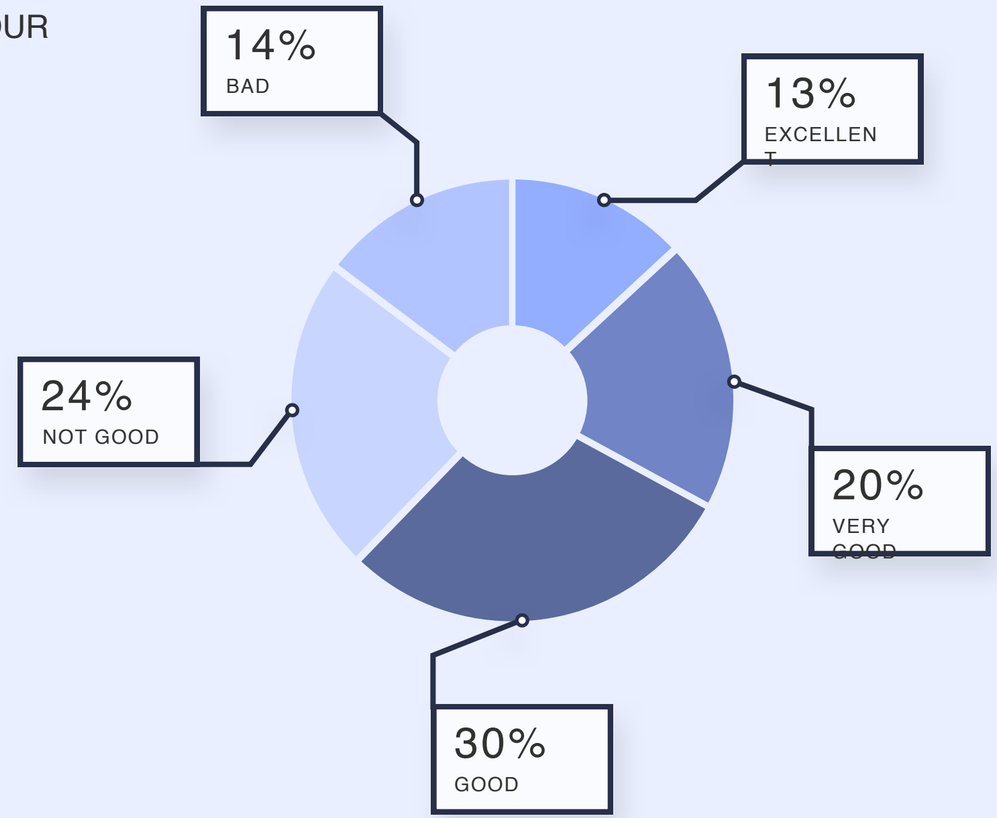
This way, QA can prepare for what's ahead, estimate the time it will take, understand what the product manager has in mind and ask the right questions to ensure the right functionality is developed throughout each stage of development.

As an organization, everyone is responsible for deploying the product in the best way possible. There needs to be a company-wide shift in mentality. When the entire

The future of testing

While most IT organizations understand the critical need and importance of DevOps, the 2018 Global Developer Report reveals few have completely adopted the methodology. A study of more than 5,000 responses from software professionals around the world, the report uncovers that although the majority believe DevOps improves cycle time, only 23% say they have adopted a DevOps workflow.

HOW WOULD YOU GRADE YOUR ORGANIZATION'S USE OF CI?



78% of managers, compared to 55% of developers, say that automating more of the SDLC is a top priority in their organization. 71% of respondents who practice DevOps also agreed automation is a hight priority, compared to 60% of respondents who practice Agile.

However, companies that are still relying on manual testing and traditional development processes are severely lagging behind. They're spending thousands of dollars and hours of productivity on bugs that could've been fixed months beforehand for a fraction of the price.

The importance of CI/CD is no longer in question — continuous testing is the way of the future. To keep up with leading companies, you have to integrate automation into your DevOps process. With skilled developers managing your CI/CD servers, you can author short, concise tests that run continuously with every commit and deploy successful changes automatically, cutting out unnecessary costs and ensuring on-time deployment.

But it's important to get everyone on board. The first step to successfully integrating automated CI/CD into your DevOps process is to communicate its value to everyone in your organization, developers and QA engineers alike. Build testing into your company culture, and then use automation to ensure this testing is occurring.

With smart, automated testing, you will increase on-time deployment, reduce the overall costs of production and increase the agility of your applications.

Quality as an Organizational Initiative

The way in which a company's R&D efforts is structured will have a direct impact on the quality of their products, services, and procedures. For software specifically, the question revolves around ownership of the development and QA process. With more tools becoming available to developers and QA testers, and with the integration of AI, companies can customize the structure in any way they see fit.

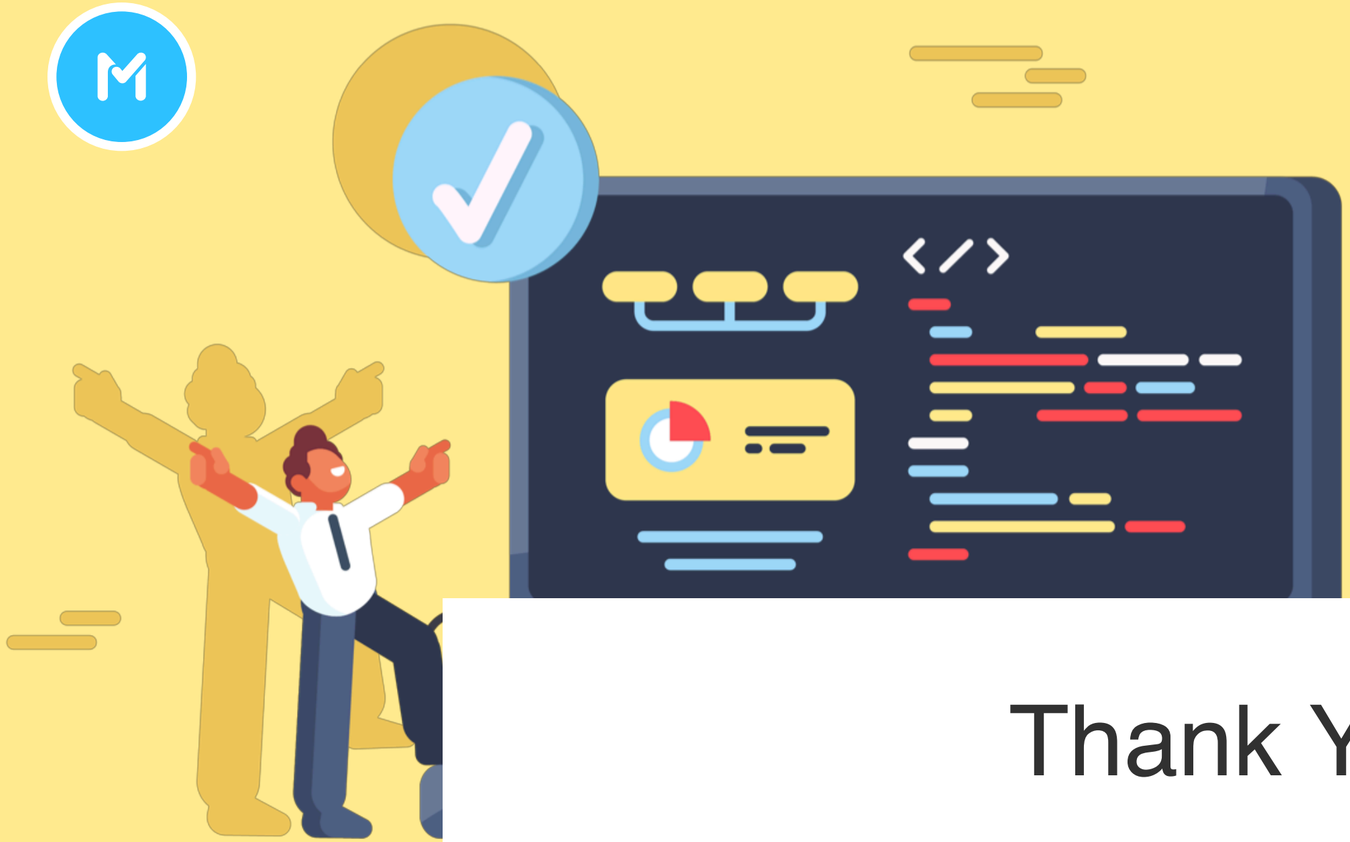
Companies like Engagio, Wix, Walmart, and NetApp are already using automated testing solutions, and they all have a unique organizational structure for their R&D activities. A solution like Testim, that makes use of artificial intelligence and allows anyone to easily author, execute and optimize tests, will bring value to any business.

About the author

Raj Subramanian is a former developer who moved to testing to focus on his passion. Raj currently works as a Developer Evangelist for Testim.io, that provides stable self-healing AI based test automation to enterprises such as Netapp, Swisscom, Wix and Autodesk. He also provides mobile training and consulting for different clients. He actively contributes to the testing community by speaking at conferences, writing articles, blogging, making videos on his youtube channel and being directly involved in various testing-related activities.

He currently resides in Chicago and can be reached at raj@testim.io and on twitter at @epsilon11.
He actively blogs on www.testim.io and his website www.rajsubra.com. His videos on testing, leadership and productivity can be found here

Have a sound testing strategy, but in need of a test automation platform to support it? Then please contact info@testim.io to schedule a demo and see how Testim integrates with your CI/CD process.



Thank You!

For more information contact us at info@testim.io

Or call +1-650 - 665 - 9293