OnPath

Automation Testing: Are you ready to start?

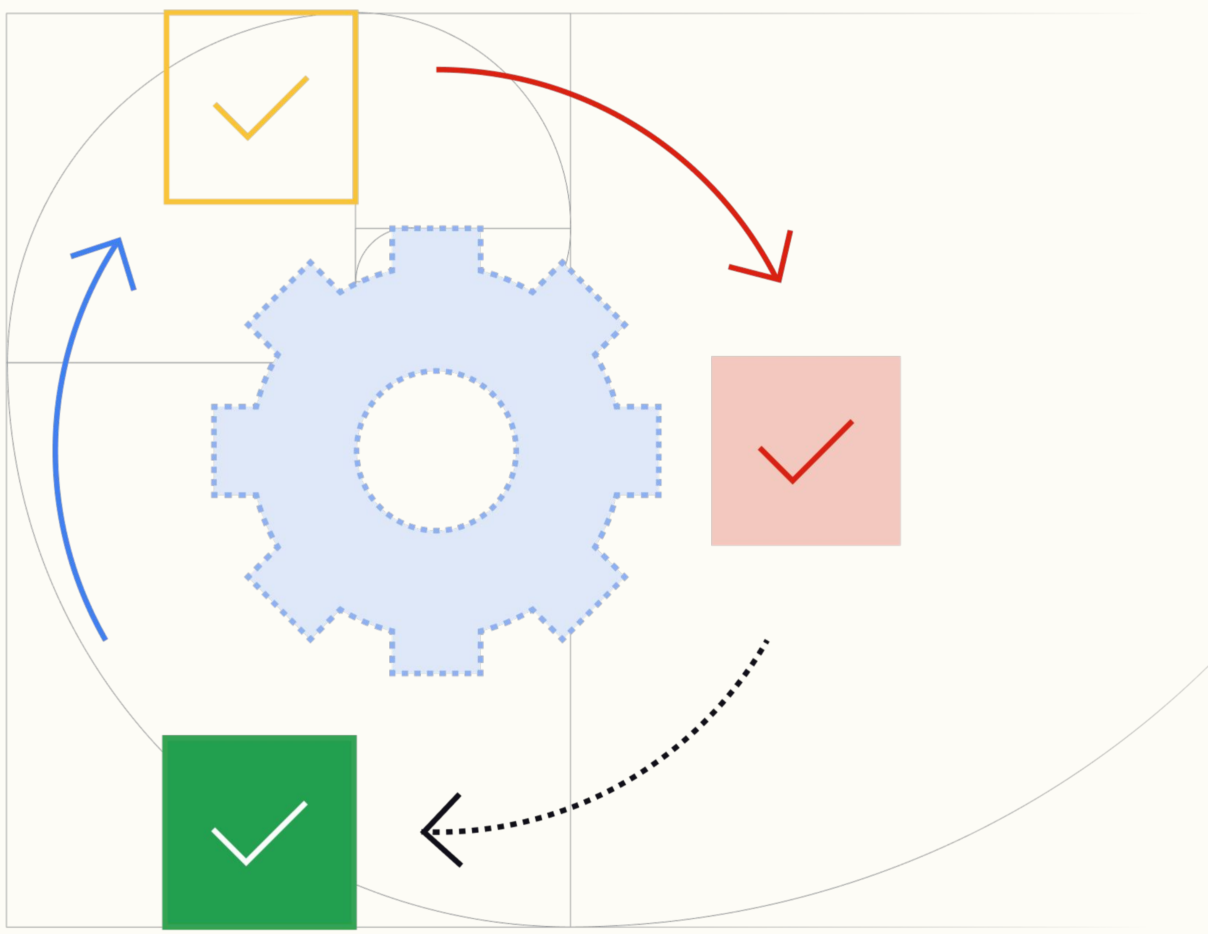
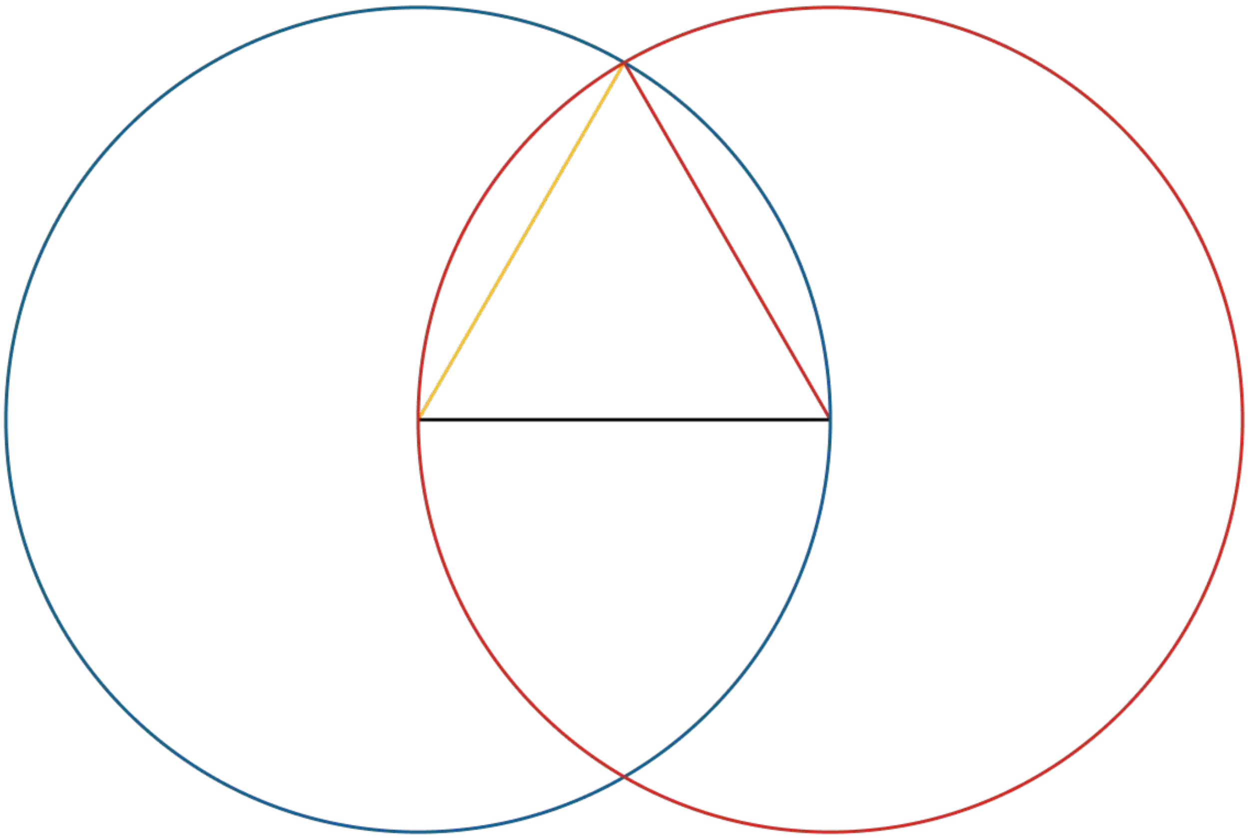


Table of Contents



Deadlines are approaching. Your QA teams are racing through manual testing phases to get the product out the door.

You're getting the job done but you're working long days (and maybe nights) as the launch approaches. You're itching to move into automation testing. You're sure it'll speed up your delivery time and further reduce errors.

Wait. Remember what Jim Hazen said, **"It's automation, not automagic."**

Before you dive in, build a clear idea of what automation will do for you and assess whether you're ready for it. From our experience at **OnPath Testing**, **there are several key factors** that will tell you if the time is right for automation testing.

Let's assess your current capacity, determine whether your QA teams are ready for automation, and guide you towards implementation.

Understanding your position

Have you got what you need for automation?

Take a look at your current QA capacity and evaluate your resources from **an automation perspective.**

Ask yourself:

• **What is my available budget?** Automation will save you money in the long run, but it takes time to get there. Most people don't realize or consider that automation development is a *development* effort, and requires the same level of commitment as any development project. There will be bugs, maintenance issues, and tech debt. In order to manage this successfully long term, be sure you are clear about costs up front. There may be tool costs, depending on your tech needs, and likely a dedicated resource or two, along with possible training costs.

• **Do we have the right processes in place?** Outline a successful test automation strategy to take full advantage of the benefits of this technology. You'll want your software development lifecycle (SDLC) to include time for automation, or make sure it's happening seamlessly, in parallel with your other QA efforts.

Carefully choose what to automate so there's no time wasted on unnecessary maintenance. Have the DevOps lead work closely with automation folks to have all the right CI/CD pipelines in place for automated execution and results reporting.

• **Have you looked into training?** If your team lacks hard skills, and additional coverage will not be available, will training be offered to get automation testing off the ground? If so, where is that training coming from? Have you considered the cost?

Be aware that automation solutions rarely come in a single package — more common is to use separate tools covering different aspects of the test automation, such as Postman for API testing, Selenium or Cypress for functional testing, and JMeter for performance testing.

The answer to these questions might be a categorical 'yes', and that's a great place to start. Be aware however, that there is always more to discover as you dig into an undertaking like this.

Take staff as an example. Your scrum master might insist you have the right people to execute automation and there is little to no lift, but have you asked the team? They might feel less confident. You may think to use your developers for automation development, but have you considered the very real need for Dev/QA separation? Listen to your staff and find out what's realistic.

Why Do You Want to Automate?

It's important to define why you want to automate. How will it benefit your project? Will it bring you closer to your goals?

Reasons to automate may include:

- **Decreasing execution time**. Automation testing allows you to execute consistent and regular test cases, saving time and effort. The result will be a reduced time-to-market and happier stakeholders.

- **Greater test coverage.** Time is no longer your enemy. Automation testing means you can test more features in the same, or less, time and with reduced impact to the team.

- **Struggling with errors.** Manual testing is susceptible to human errors and that's lengthening your testing phases. Automation testing will accelerate this process and improve your overall accuracy for repetitive tests.

- **Re-using the test suite.** As you build your automation suite, these tests become part of a growing library of granular components that may be put together for specific test plans or user scenarios.

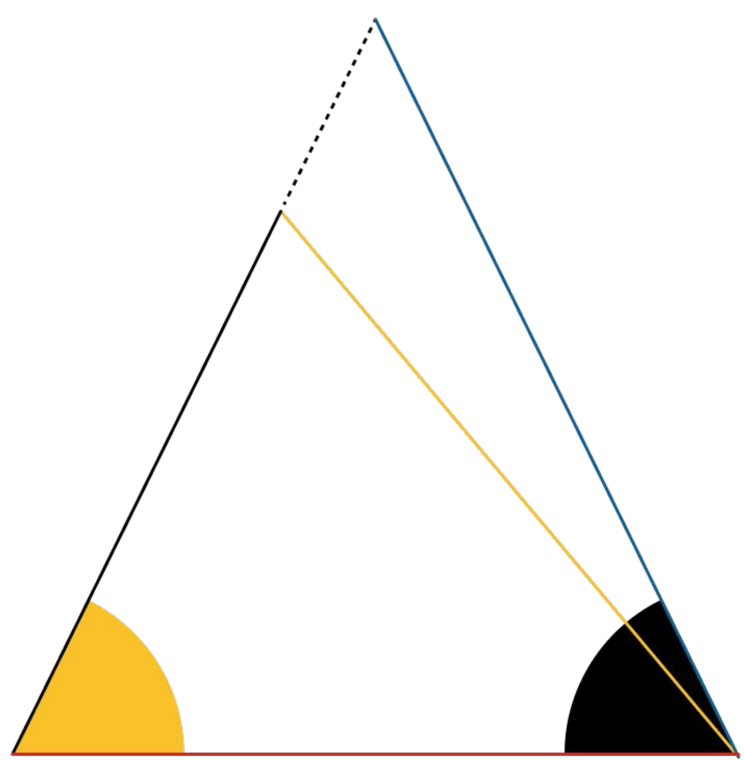
Why do you want to automate?

Identify the business needs. What tests are repetitive yet frequently performed? What business-critical functionality must always work? What are the demo scenarios that support your sales team or C-level execs?

- **Functional testing.** An automation tool won't skip any steps, and ensures that data is always input as desired. It will also enable sequential test runs with different data to compare the results in a shorter timespan than manual testing. Specific types of testing, such as accessibility testing, lend themselves to automation, and can be crafted in a way that maintenance is minimal.

- **Regression testing.** The most common, and best target of automation testing, is on stable code that doesn't change frequently. These mature parts of your application can be covered by automation and rarely touched, providing the holy grail of regression test speed and coverage that all tech managers strive for in each release.

- **Performance testing.** Automated performance tests will keep stakeholders happy as it ensures any performance spikes will be managed in gracefully.



Considerations before implementation

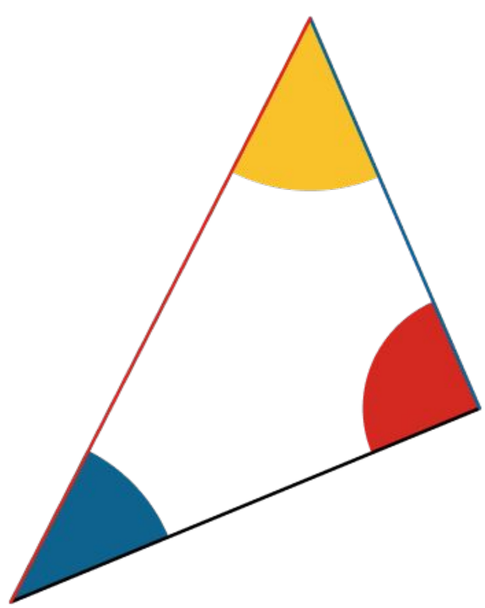
You've gone through the questions. You've got a strong understanding of your current position. You know you're ready for automation testing. You know why you want to automate, and you know what you want to automate.

What's next?

There are a few things **to consider** as you move towards implementation:

▲ Choosing the Right Tool for Your Tech Stack

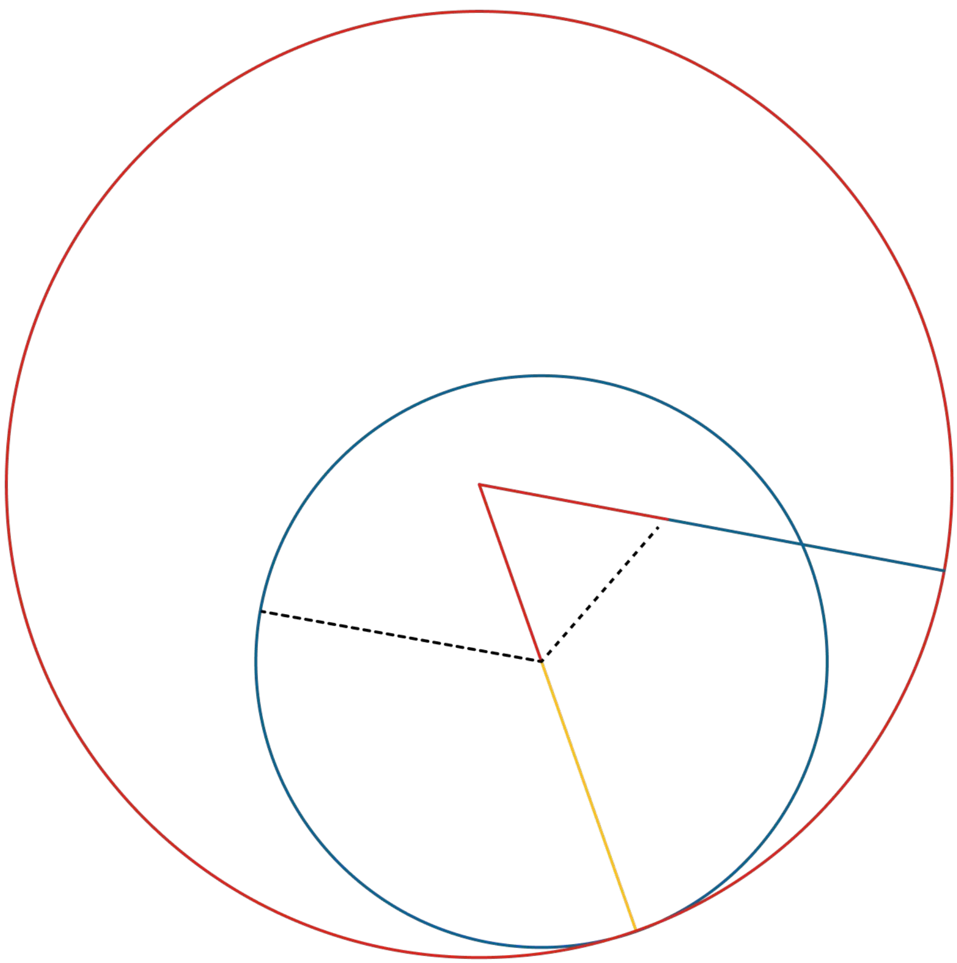
There are a plethora of automation tools and frameworks in the marketplace, and more and more utilize AI to meet end goals. Low-code and no-code options include standard platforms like Selenium, Cypress, and Nightwatch. Different frameworks perform better than others with specific technologies. Some are mobile friendly, some aren't. Take the time to have your lead engineer research the top choices, examine pitfalls, and present options, so that once you take the leap, you're not changing course sooner than expected.



Is Your Primary Testing Tool All You Need?

Remember — there are many ways to automate testing requirements, including functional and API testing and security and performance to name just a few. OnPath recommends starting with a clear focus in mind — say you have a massive API library, so it's critical that you test each endpoint. Or perhaps that isn't nearly as important as a need to stay compliant with the WCAG accessibility standard. Whatever it is, keep a focused implementation as your highest testing priority and add to it over time.

Keep in mind that because different automation tools focus on different software testing needs, you will eventually have a tech stack of automation testing tools, just like development-side tech stacks. For example, it's not uncommon in a mature automation environment to see Selenium or Cypress used for functional automation, Postman for API automation, JMeter for performance testing, Appium for mobile devices, Browserstack for cross-browser testing, etc.



Is your testing environment stable?

Maximizing the benefit of automated testing requires stable a application and environment. The entire application doesn't have to be polished, but the feature set that you wish to automate should not be undergoing major changes. Functionality in flux causes unnecessary rework as your automation engineer constantly tweaks their tests to keep up with the changes, so you aren't bombarded by false positive results. Remember: each failure reported by your automated execution run must be examined by a human to determine what happened, that the bug is reproducible, and that it's not a false positive and actually an error in the automation script.

False alarms in pass/fail results can be a killer for an automation effort — too many false positives not kept in check or ignored ruin the result. Too many false negatives means bugs are being missed, which negates the entire purpose for automating tests.

External factors that may affect stability include:

- Code issues — poor coding practices, too much tech debt, etc.
- Issues with the system architecture
- Suboptimal test environments - performance, network, etc.

These factors insert issues into your testing effort, creating inaccurate results and anomalies.

Is Your DevOps Team Ready for Automation?

QA isn't the only team affected by test automation. As previously mentioned, to get the most from this effort, the DevOps lead must be ready to utilize these tests within a CI/CD pipeline. Every time a developer creates a pull request (PR) to merge their code back into the repository, a suite of tests can be automatically kicked off to produce immediate feedback if their code is not up to snuff. Utilize this important facet of automation, otherwise you're not getting the full bang for the buck.

Reporting and metrics

What Equals Success?

What equals success? What's important for you to measure?
Typical metrics to capture may include:

- **Total test duration.** How long are tests taking? Is the difference in time between manual and automation negligible or significant? Does it offer any additional opportunities?
- **Percentage of tests passed or failed.** This is a simple, but valuable metric to track testing progress across each iteration or release. It's also an easy metric to convert to graphs and charts—detailing the tests that passed, those that failed, and tests yet to run.
- **Number of defects.** Knowing the number of defects in your tests can help you with predictive modeling by indicating what other issues are left to dig up and how different testing environments affect the outcome.
- **Coverage amount.** Knowing how much of the complete functional or regression test suite is covered by automation is helpful, and should be continuously increasing.

Make sure you have a baseline comparison for each metric. You'll know if your automation tests are having the desired effect or making anything more difficult.

Future maintenance

Understand the maintenance costs in your implementation.

Build your automation framework with the goal of minimizing maintenance. Keep the code in your automation scripts streamlined. Avoid repeating code and write only what's necessary to complete the task. Keep it simple where applicable.

Remember, automation isn't everything

If things aren't working out, don't panic. There are many other avenues to get you to your QA testing goals.

At OnPath, we've found that stopping to consider these points before leaping into automation is invaluable. This exercise helps you understand why you're implementing automation testing and ensures you're supporting your goals.

We love to hear about projects. If you want to talk about yours, feel free to schedule a conversation with us!

OnPath

About Onpath Testing

For more than 20 years, OnPath Testing, a Colorado-based, award winning QA service, has partnered with organizations in a wide range of industries to ensure their software works as expected and as designed.

OnPath has led the way in establishing a QAOps model within dev teams, and has taken a forward stance on IoT and AI-based tools and methods. OnPath champions WCAG digital accessibility standards in theory and in practice as we move into a new era of connectivity.

We love to talk about projects. Feel free to get in touch.
get in contact today.

Contact us