WHITEPAPER

AUTOMATION ESSENTIALS FOR THE AGE OF AGILE

APPLAUSE

> The first rule of any technology used in a business is that automation applied to an efficient operation will magnify the efficiency. The second is that automation applied to an inefficient operation will magnify the inefficiency."
>
> **BILL GATES,**
> CO-FOUNDER, MICROSOFT

No enterprise in the age of Agile can afford to ignore test automation. The method's popularity grew by 85% across all industries from 2015 to 2017, but many companies are still lagging behind.

If your organization is seeking to benefit from test automation, careful planning and preparation are essential.

This whitepaper provides a guide to starting with automation, and introduces six essentials for succeeding:

1. **Evaluating** your testing strategy
2. **Introducing** automation into your deployment pipeline incrementally
3. **Aggregating** all test results to support informed decision making
4. **Establishing** the right framework
5. **Handling** failed automation scenarios as critical failures
6. **Striking** the proper balance between automated and manual testing

How much can your organization hope to benefit from test automation? As Bill Gates suggests, it is no cure-all. Rather, it can help your organization leverage existing operational efficiencies to maximum benefit.

Evaluating Your Testing Strategy

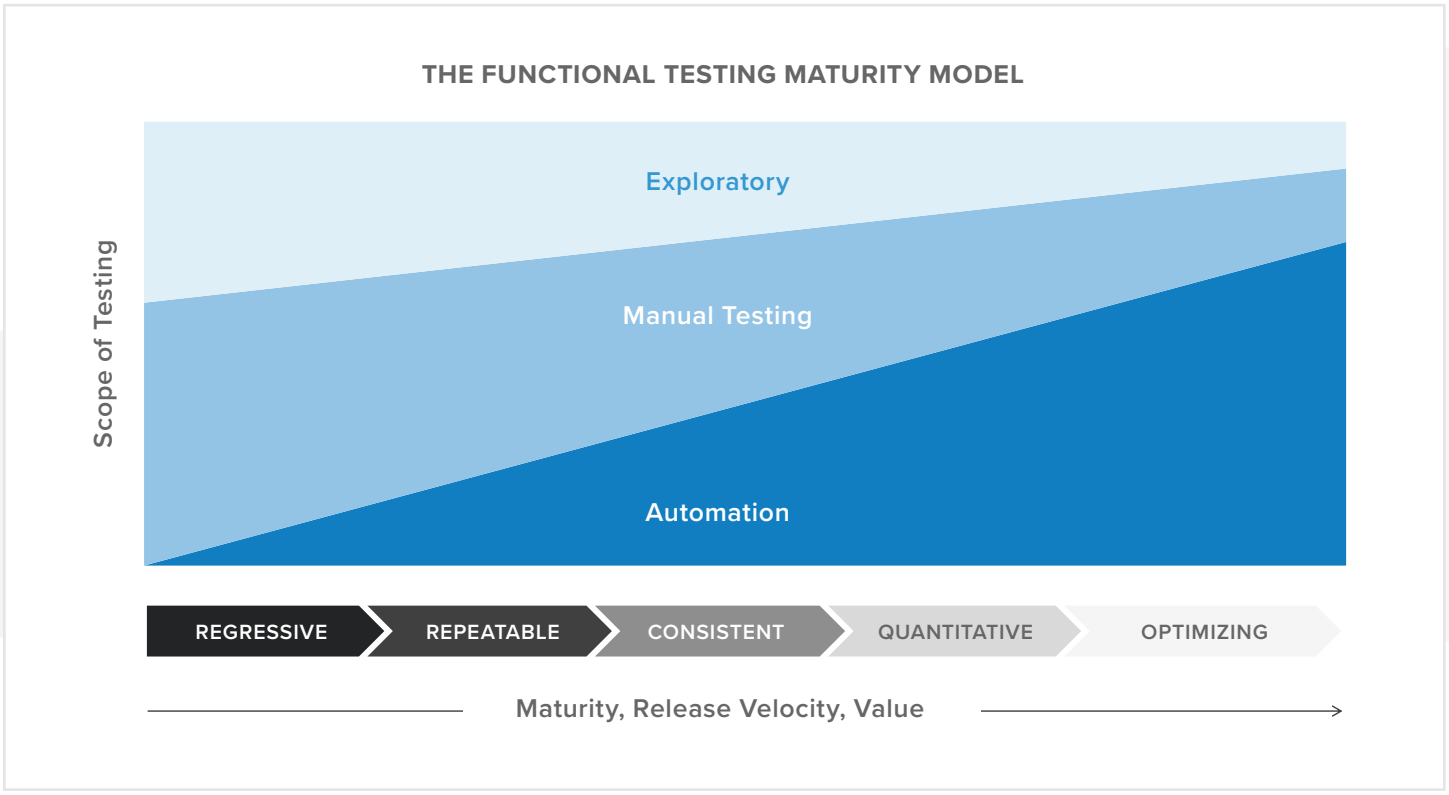
Before making a move toward increased automation, evaluate where your organization currently stands with its global approach to testing. The goal is to devise an automation strategy that aligns with your broader testing strategy and fully leverages the unique strengths of your people and corporate culture.

Your evaluation should focus primarily upon three key areas. Use the below questions to evaluate your current process. Doing so will help to ensure a smooth integration of automation.

1. **Team:** What is your team makeup? Where are the expertise gaps? Are QA and Development teams working in silos, largely isolated from one another?
2. **Technology:** How will automated tests be triaged? How is automation integrated with a bug tracking system?
3. **Process:** When are tests written? How are tests updated? How long are your sprint cycles? How often do you release? How does feedback guide your testing strategy?

Rushing through this process may ultimately prove detrimental to the efficiency of your organization's software development capabilities.

Your automation journey should sync with your maturity level. This will allow you to devise and execute an effective strategy for deploying automation. Based on the maturity of the application, it should undergo an appropriate blend of exploratory, manual, and automated testing.



Areas of the application that are more mature or change less frequently are the best candidates for implementing test automation. Conversely, newer and more fluid areas should undergo greater amounts of exploratory and manual testing.

An organization that skews toward the regressive on the maturity model scale, for example, will be best served by focusing more on documenting, refining and streamlining processes, and less on automation.

As you mature your testing strategy, the rate at which you are able to verify changes within your application will increase in velocity, allowing you to validate the application for faster release and potentially decrease your time to market. This isn't just true with automation, but with manual testing as well. By automating the right things, you'll also be able to focus exploratory and manual test case execution efforts on the most critical or complex parts of your code.

Introducing Automation into the Deployment Pipeline Incrementally

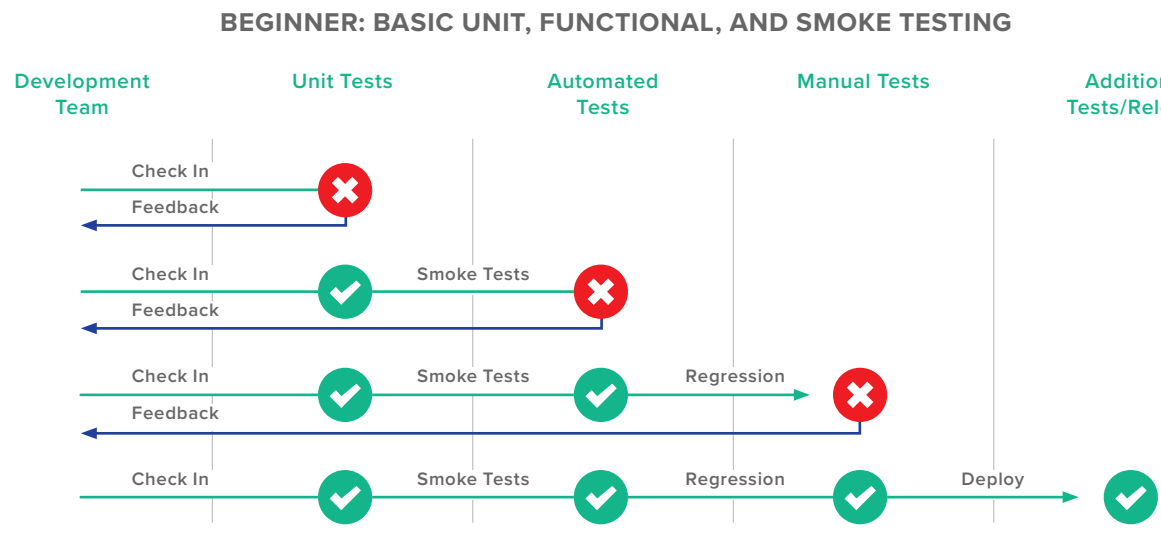
There are three distinct test automation stages to help you introduce automation into your deployment pipeline incrementally.

BEGINNER STAGE

This stage is all about creating quick wins for your team, boosting both morale and enthusiasm for the journey to automation. Ensure unit and functional tests are run on every build and the reporting is clear and correct. You should also have a way to measure the code coverage of tests.

Once the unit tests are consistently passing, the next step is to implement a very small set of automated smoke tests. Initially, it may be easier to run these tests at night. But eventually, you'll want these tests running as an integral component of the build process; they should be triggered automatically upon the completion of a successful build. Once these tests are passing consistently, you can begin adding more smoke tests to your automation suite.

The progression of adding only one small test at a time will help your organization to quickly build confidence in automation, and will serve as the foundational building blocks for your automation process. This incremental procedure will also help to prevent the major losses of time that can result from having to troubleshoot and fix a larger set of tests.



(Start with automating simple smoke tests in the Beginner Stage while still doing manual regression testing.)

INTERMEDIATE STAGE

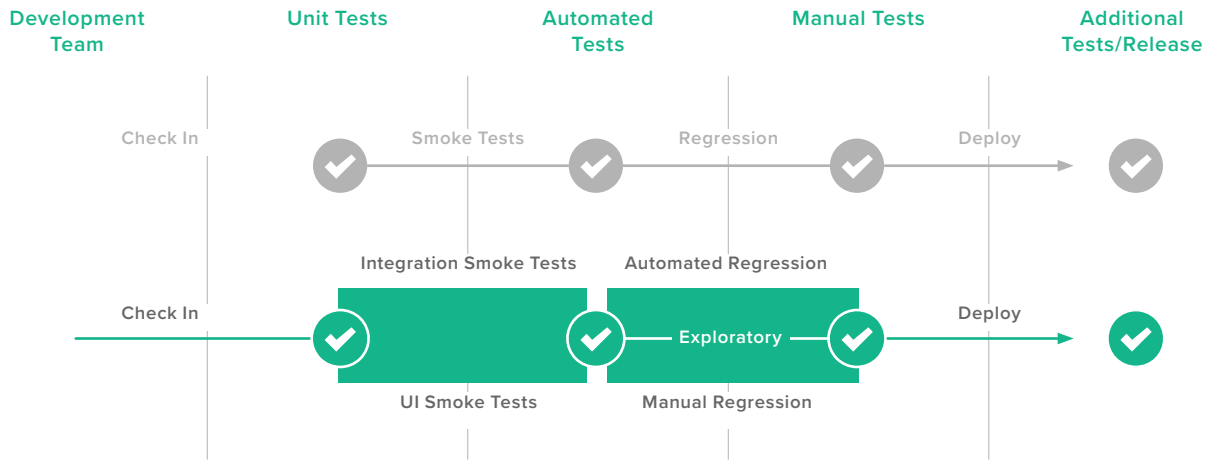
Now that you have built some confidence in your initial set of smoke sets, you can begin to attack additional layers of the application. For example, if you have an application that enables execution of both integrated and UI tests, this would be a great opportunity to expand your tests and run them in parallel. The more you can cover in the functional testing pyramid in this manner, the greater your confidence will be as you proceed with your regression testing.

Regression testing is where automation can really begin to pay significant dividends. You'll be able to run multiple layers in parallel, decreasing the time required to obtain test results and feedback on the quality of the application. And, just as with automated smoke tests, your automated regression tests will grow incrementally until you've reached a comfortable level of test coverage and can begin optimizing the process.

In conjunction with your automated regression tests, your manual team should prioritize the execution of structured manual tests along with ongoing manual exploratory tests. The greater the percentage of regression tests you can automate, the more time your manual testers can devote to exploratory testing. This will enable increased coverage of edge cases—where the more interesting and problematic bugs often hide.

The primary goal of this intermediate SDLC stage should be to establish a test case management system and reporting structure. Storing all test results in a single repository that accommodates a single, catch-all view will provide more clarity about what to, and what not to, deploy.

INTERMEDIATE: UNIT, FUNCTIONAL, SMOKE AND UI, AND REGRESSION TESTING

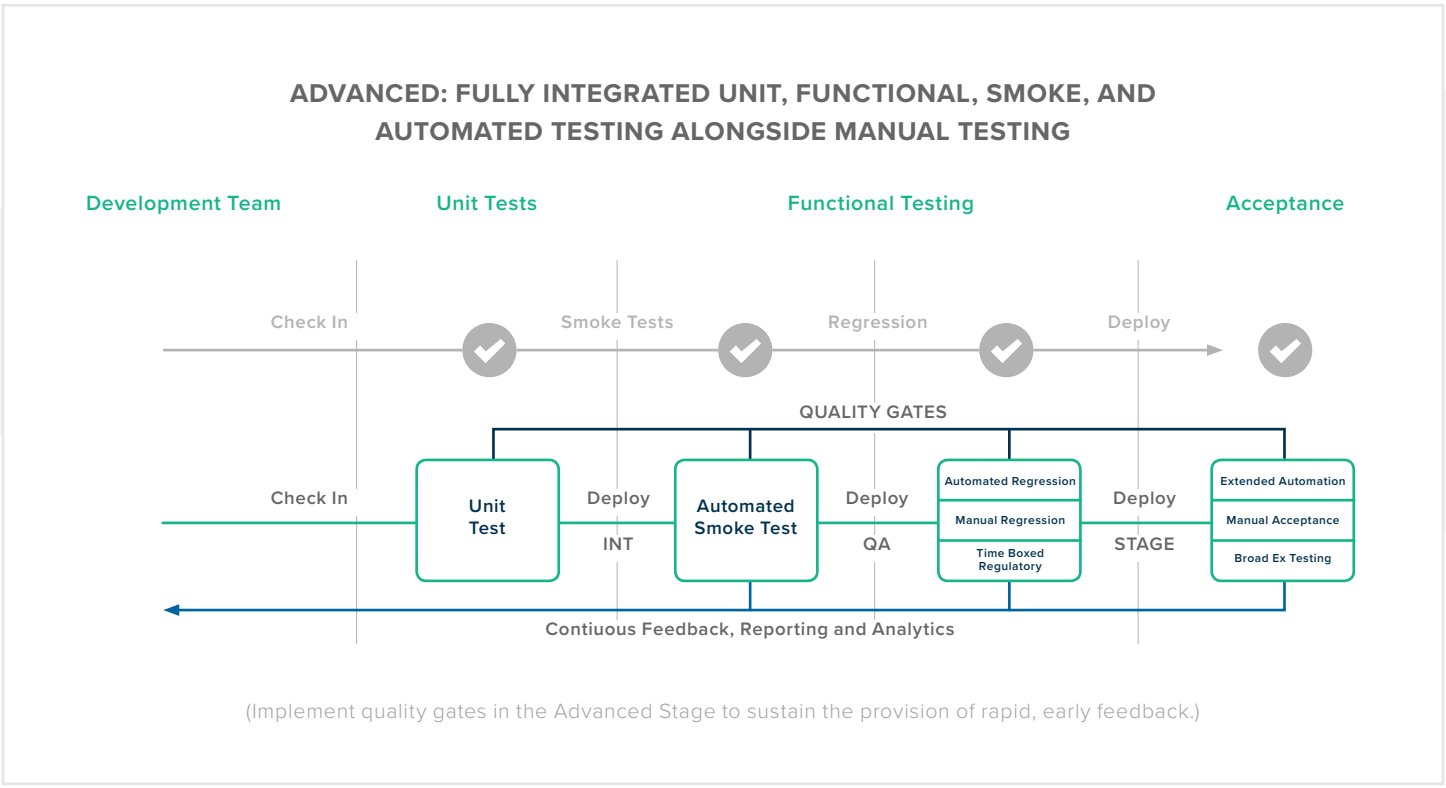


(In the Intermediate Stage, you are ready to start automated regression testing on top of automated smoke tests.)

ADVANCED STAGE

The greatest difference between the intermediate and advanced stages is found in the implementation of quality gates. Aligning with your need for a test case management system, quality gates will facilitate the rejection, as necessary, of builds before deployment or before moving on to further testing.

Your goal in this stage is to sustain the provision of rapid, early feedback to your development team to support timely root cause analysis. This helps ensure critical bugs are eliminated early in the development process, eliminating more costly and timely fixes down the road.



(Implement quality gates in the Advanced Stage to sustain the provision of rapid, early feedback.)

Throughout the process of introducing automation, the key to success is to perform 'just enough' testing at each phase before you expand the depth and breadth of your automation. This approach establishes a point-of-comfort before adding new test environments, providing quicker feedback to your development team.

Finally, you'll want to build confidence at each quality gate before running more time-consuming, costly test automation. Running those time-consuming tests often creates bottlenecks that slow the development process. Pushing the more complex automated testing further down the road can ultimately help to reduce delays throughout the entire development process.

Aggregating All Test Results to Support Informed Decision Making

Providing a strong ROI enables key stakeholders to make data-driven decisions based upon measurements of application quality. This capability supports the primary goal of releasing high-quality applications that consistently delight end users. Important factors include:

- Test Analytics
- Performance Metrics
- Test Result Reporting
- Insights
- Automatic Alerts

When provisioning for automation, remember you will need to make a number of technology investments. Be certain these align with your target ROI. You will need a framework with the appropriate features and functions built into it to run in a fast and stable fashion, making it easy to find and solve issues.

Establishing the Right Framework

The key enhancements needed in such a framework can be broken down into the following pieces:

- **Architectural Patterns** such as the Page Object Model and the Page Factory provide the structure to create a robust automation framework—one that follows patterns already proven to solve common issues.
- **Error Handling** provides a key enhancement that is necessary in a solid automation framework. The open source libraries used to connect to drivers and applications have come far in recent years. Even so, there are always errors that must be handled to ensure smooth and robust execution of automated scripts.
- **Device Cloud Integration** is essential in providing a 'place' to execute automation.
- **Integration with a CI Server** is preferred in the execution of the test package as it allows you to move towards a continuous testing and delivery model.
- **Test Case Management Systems** are used to automatically tie coded tests to human readable tests, and also aggregate automated test results alongside manual test results for easy analysis of application health and build/release stages.
- **Test Data Management Systems** are commonly built to store all the test credentials, expected conditions, localized translations, etc.
- **Bug Tracking Systems** should be tied to failures in your test case management (and vice versa) for easy root cause analysis and tracking of issues at scale.

The quest to improve ROI within any software development organization might be somewhat likened to a treasure hunt—and like the best treasure hunts, it's a quest that offers great potential. According to Hobson & Company research, "Software failures impacted an estimated $1.7 trillion in assets in 2017, and more than 70% of those failures were the direct result of software bugs or usability glitches."

HOBSON & COMPANY'S RESEARCH FOUND, ON AVERAGE, APPLAUSE CLIENTS ENJOYED:

150-200%

increase in the capacity of internal testing teams

100-150%

increase in the number of annual releases

50-75%

reduction in testing lab costs

Handling Failed Automation Scenarios as Critical Failures

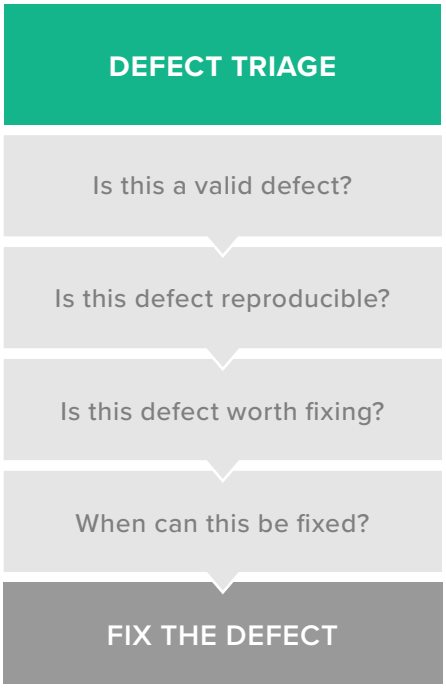
A number of procedures commonly occur before the software build; following a shared coding standard across the organization, for example, and implementing code review procedures so new changes are reviewed and commented upon (static testing) before merging. Development should be responsible for unit tests, and these tests should all pass before merging.

Once changes have been merged, integrating the build process with automated code coverage, continuous feedback, and quality gates for coordinating manual verification are necessary.

Passing tests should provide all stakeholders with assurance and confidence that the application is functioning as intended, and it is ready for use by customers. Ignoring failures serves to erode user trust while increasing the risks that may be incurred post-release. In fact, the best practice is to treat each failing test as a release blocker. If there are flaky tests eroding confidence, remove them from your automation suite, run the tests manually and refactor them. After all, if you can't trust the test results, the tests are providing negative value.

Another best practice? Triage automation test results just as you would bugs found through manual testing. Consider there can be multiple outcomes for failing automation scenarios; a failure can result from a product bug, a test automation failure, a bad test case, or any of multiple other issues.

Instituting an intermediary triage stage will save your engineering team time in the long run. There's no need to treat your processes for automation results any differently than you would bugs found manually. Access to materials like device logs and screenshots, and even videos of the test run, will only help to strengthen and speed up the triage process.



Striking the Proper Balance Between Automated and Manual Testing

Automation is not a silver bullet solution. And manual testing will remain a necessity far into the foreseeable future. The challenge, then, lies in striking the proper balance between automation and manual testing.

When an imbalance exists between automated and manual testing, resulting problems can include:

- Bottlenecks in the development pipeline
- Untenable automation suite
- Poor ROI

There is, however, no hard-and-fast formula for determining the right balance between automated and manual testing for your organization. That balance is different and unique for every software development organization, and is influenced by the maturity of your testing process.

How can you find the right balance for your organization?

You must carefully evaluate your testing strategy and your team to determine where, and to what depth, both automation and manual testing fit in your organization.

LEARN MORE

ABOUT APPLAUSE

Applause is the worldwide leader in crowdtesting and digital quality. Software is at the heart of how all brands engage users, and digital experiences must work flawlessly everywhere. With 400,000+ testers available on-demand around the globe, Applause provides brands with a full suite of testing and feedback capabilities. This approach drastically improves testing coverage, eliminates the limitations of offshoring and traditional QA labs, and speeds time-to-market for websites, mobile apps, IoT, and in-store experiences.

Thousands of leading companies — including Ford, Fox, Google, and Dow Jones — rely on Applause as a best practice to deliver high-quality digital experiences that customers love.

Learn more at www.applause.com.

NORTH AMERICA
100 Pennsylvania Avenue
Framingham, MA 01701
1.844.300.2777

EUROPE
Obentrautstr. 72
10963 Berlin, Germany
+49.30.57700400

ISRAEL
10 HaMenofim Street
Herzliya, Israel 4672561
+972.74.757.1300

APPLAUSE