TestRail

The QA Lead's Guide to Agile Testing

CONTENTS

Introduction

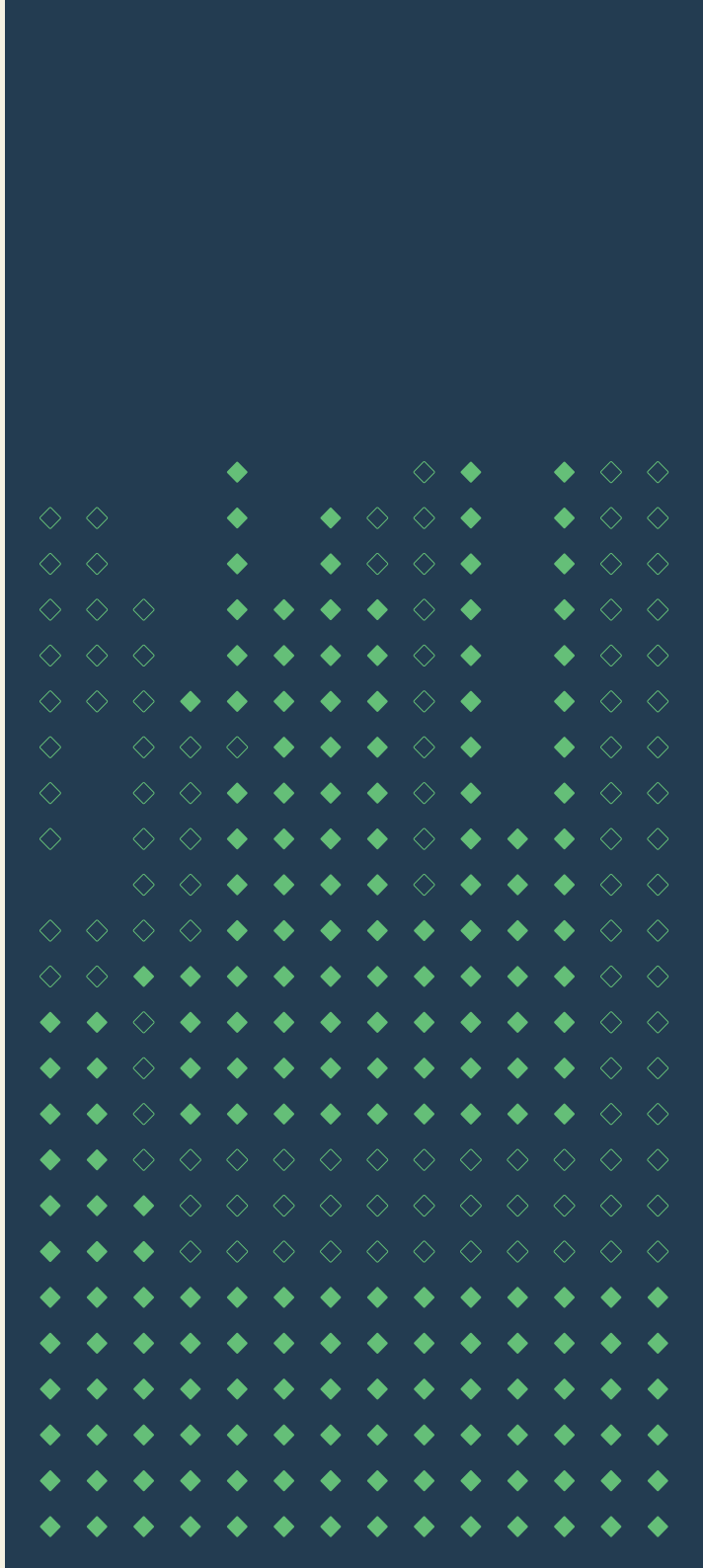
Your team is likely already utilizing Agile or an Agile-like scrum methodology to some extent; in our 2023 Software Testing & Quality Report, 73% of survey respondents reported using Agile in their testing program, and 55% reported using Scrum. However, despite such widespread adoption, many organizations are still working toward getting Agile “right” and finding the balance between fast release cycles and shipping high quality software.

Agile also poses a unique set of challenges for testing. With the need to have shippable code produced at the end of every timeboxed sprint, agile development requires testing and development to be happening simultaneously.

The nature of continuous improvement and iterations also means that requirements will frequently change, and your testing must be able to pivot to meet rapidly-evolving needs. Additionally, these frequent iterations mean an increased demand for regression testing within your testing program—which, when coupled with faster testing cycles, can lead to sparse test documentation.

This guide is intended to provide actionable takeaways in seven key areas of agile testing that any QA manager will be able to use to improve and optimize their agile strategy. We hope the tips and insights in this guide will help you on your agile journey to release higher quality software, faster.

In the interest of iterative improvement, we’ve ended each section of this guide with a “CRAWL/WALK/RUN” section. These outline the impact and improvements you should expect from your team when first implementing the concept (CRAWL), when you are getting comfortable with it (WALK), and when your team is fully onboarded and utilizing it in its fullest capacity (RUN).



Section 01

Get Aligned on QA's Role in Agile

Get Aligned on QA's Role in Agile

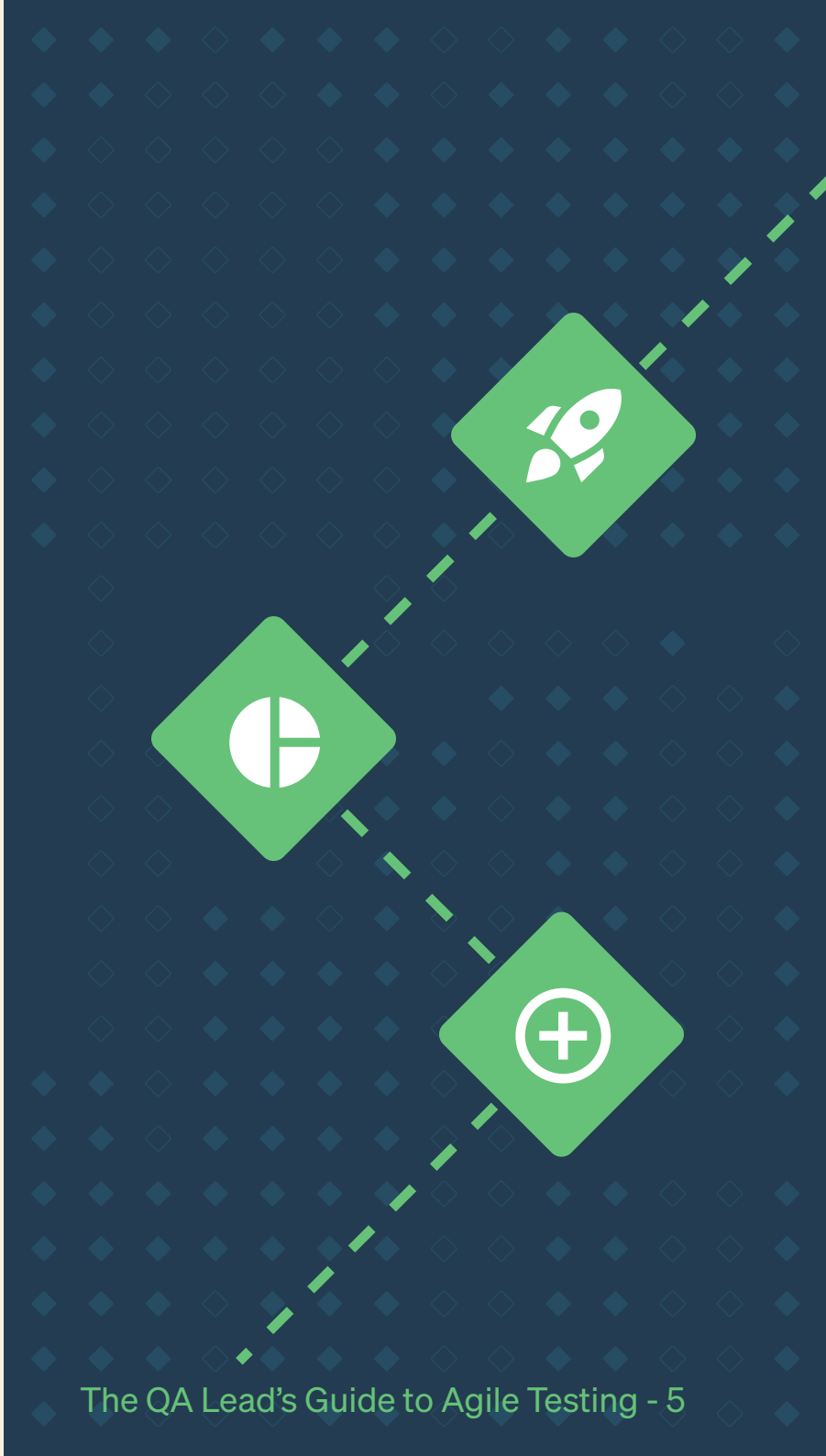
Make quality a whole-team effort

Agile methodology is just as much of a culture and communication approach as it is a technical framework.

What does this mean for your agile team?

- **Everybody** needs to have a quality-driven thought process
- The whole team should analyze performance impacts and potential risks of new changes and features **together**
- The whole team needs to foster a culture of constant collaboration—with testers included in **all discussions**

A whole-team approach is the best way to overcome the strain on time and resources inherent with agile methodology. It's important to shift your team's mindset to "development = coding+testing." When you break down the development/testing divide, it becomes much easier to accomplish everything that needs to be done within an agile sprint.



Get Aligned on QA's Role in Agile

What is the role of QA in an agile environment?

Shift your QA mindset: testers are not just testers; testers are quality coaches

For your agile testing program to be scalable, testers need to do more than just testing:

- Develop and optimize testing processes
- Help foster a whole-team approach
- Teach the team how to test effectively
- Coach on good quality practices
- Advise on strategy for automation, testability, test data, testing within the CI/CD pipeline, etc.

Get Aligned on QA's Role in Agile

01

CRAWL

Your entire team (development and QA) is on board with the agile team approach and committed to breaking down silos and embracing quality as a whole-team effort.

02

WALK

Your testers are not just testing, but establishing/improving processes and acting as quality coaches for the whole team. Developers are getting comfortable with testing their own code, and testers are being involved throughout the development process.

03

RUN

The entire team is dedicating their time and unique skill sets to testing. Testing approaches and priorities are clearly defined for each sprint. Your team has well-defined and documented testing processes, and a system in place for further optimization. Testing and quality goals are being met reliably and sustainably each sprint.

Section 02

Adapt Your Test Approach to an Agile Environment

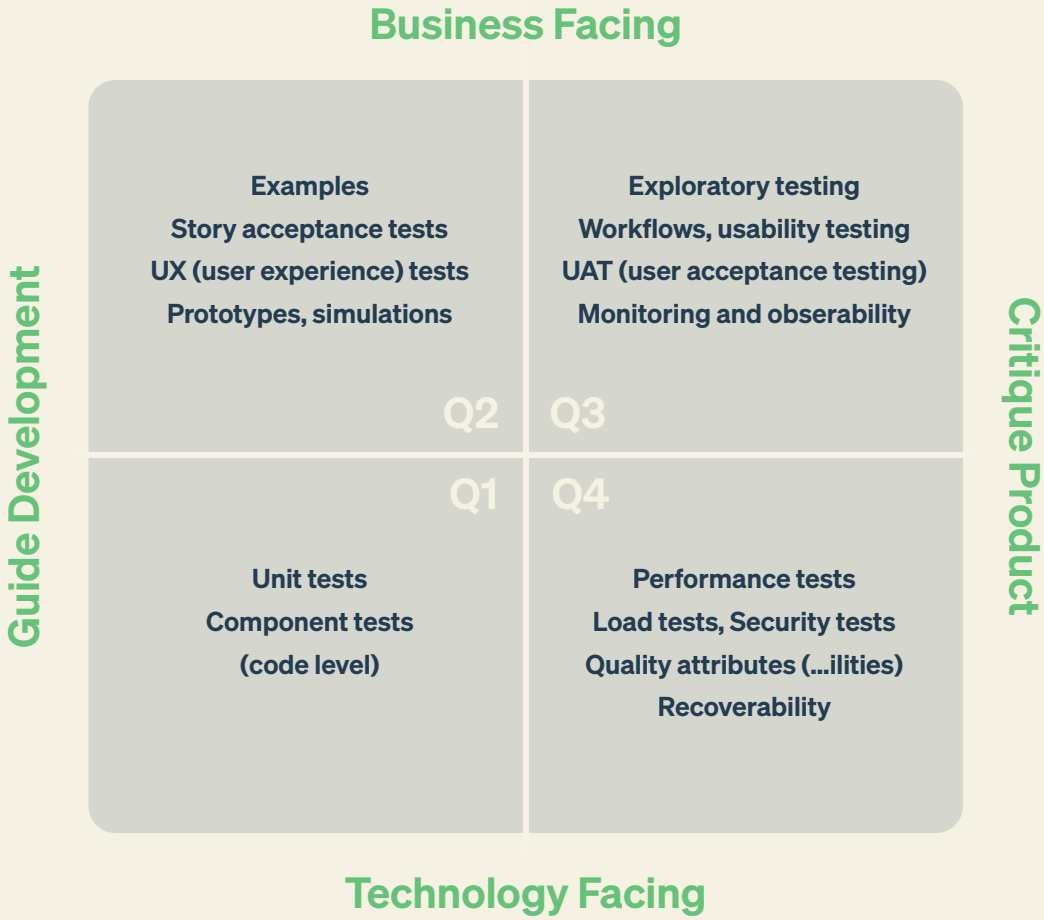
Adapt Your Test Approach to an Agile Environment

The **Agile Testing Quadrants** help guide you to the type of testing you should do in any given sprint depending on the context of your project deliverables

First, determine if the work being produced is more **business-facing** or **technology-facing**

Then, determine if the testing is meant to **guide development** or **critique the product** based on the stage you are at in your development cycle or sprint

The quadrant you land in provides guidance toward what type(s) of testing you should perform that sprint

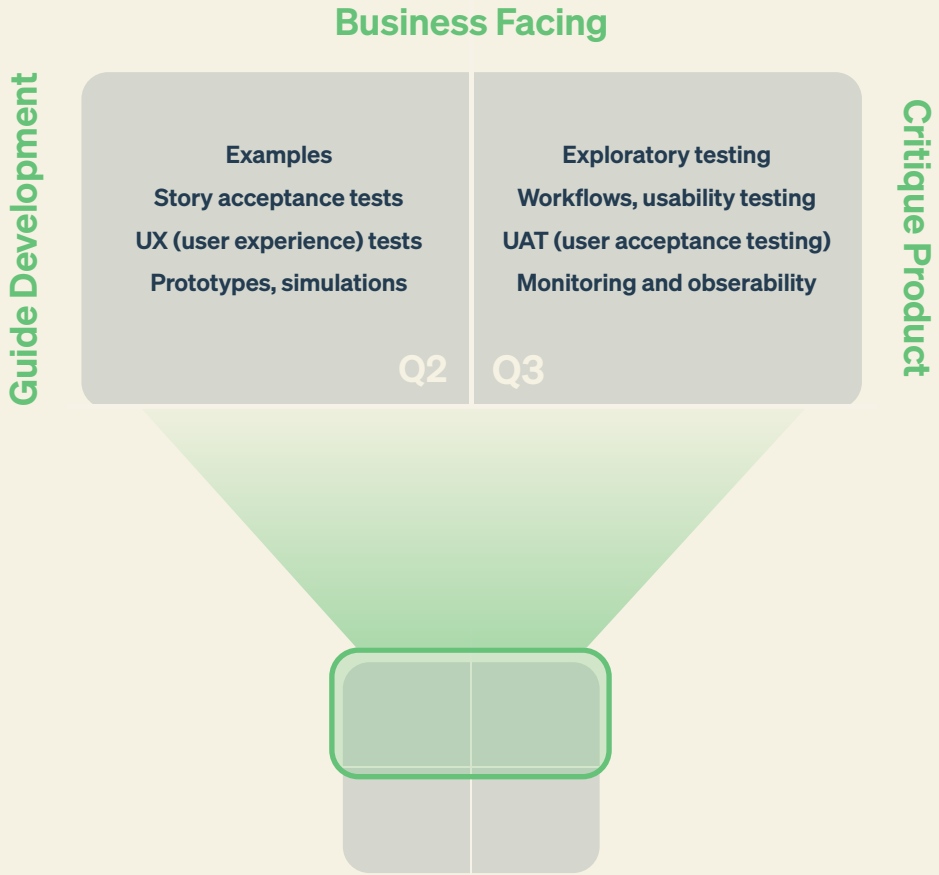


The Agile Testing Quadrants were developed by Janet Gregory and Lisa Crispin, based off of Brian Marick's 'Marick Test Matrix'

Adapt Your Test Approach to an Agile Environment

An example scenario using the Agile Testing Quadrants

You're launching a new product, and you're still in the process of validating before creating a more consistent solution. This is business facing, as the product is nearing launch, and calls for testing that will both help guide development as well as critique the existing product. For this scenario, you'd invest more in the top two quadrants.



Why use the Agile Testing Quadrants?

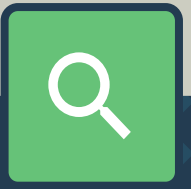
- Provides a common language for your team to think through testing approach together
- Emphasizes Whole Team approach and highlights importance of involving QA throughout the development process
- Allows you to plan testing for different levels of "done" (i.e. story done, feature done, release done) and for every step of the deployment pipeline

Adapt Your Test Approach to an Agile Environment

01

CRAWL

Your team is introduced to the Agile Testing Quadrants and is familiar with how the tool can be utilized to plan and prioritize testing.



02

WALK

Your team is adopting the Quadrants as a common language to help guide their agile testing approach.

03

RUN

Your team is consistently and efficiently utilizing the Agile Testing Quadrants for each sprint and each step of the development cycle. They are now able to plan more strategic testing approaches faster.

Section 03

Make Your Testing Process Transparent and Visible to the Whole Team

Make Your Testing Process Transparent and Visible to the Whole Team

Transparency and visibility are core concepts in Agile. With your whole team working together towards both development and testing—in timeboxed sprints, with rapidly evolving requirements—ensuring everyone has easy access to up-to-date results and information is critical.

The easiest way to accomplish this is by centralizing your QA processes with a **test management platform.**

Why use a test management platform for agile testing?

- Provides a single source of truth that is accessible to your entire team
- Integrates with project management tools such as Jira (and can even provide two-way visibility)
- Integrates with automation platforms to collect both manual and automated test results in one place
- Makes it easy to view test results, generate reports, and track coverage and traceability

Make Your Testing Process Transparent and Visible to the Whole Team

However, not all test management platforms are created equal—ensure you consider your options and choose one that integrates with the tools your team is already using, works for your budget, and provides the features that are a priority to your workflow.

TestRail is a great test management solution for agile testing teams. Try it for yourself and see first-hand how it can integrate with your existing processes and centralize your testing efforts with a 30-day free trial.

Try TestRail

Make Your Testing Process Transparent and Visible to the Whole Team

Quality gates: how do you define “done?”

It’s not enough to have all of your testing data visible to the entire team—your team must also have a common understanding of the definitions of “ready” and “done” to ensure that each sprint’s goals are completed.

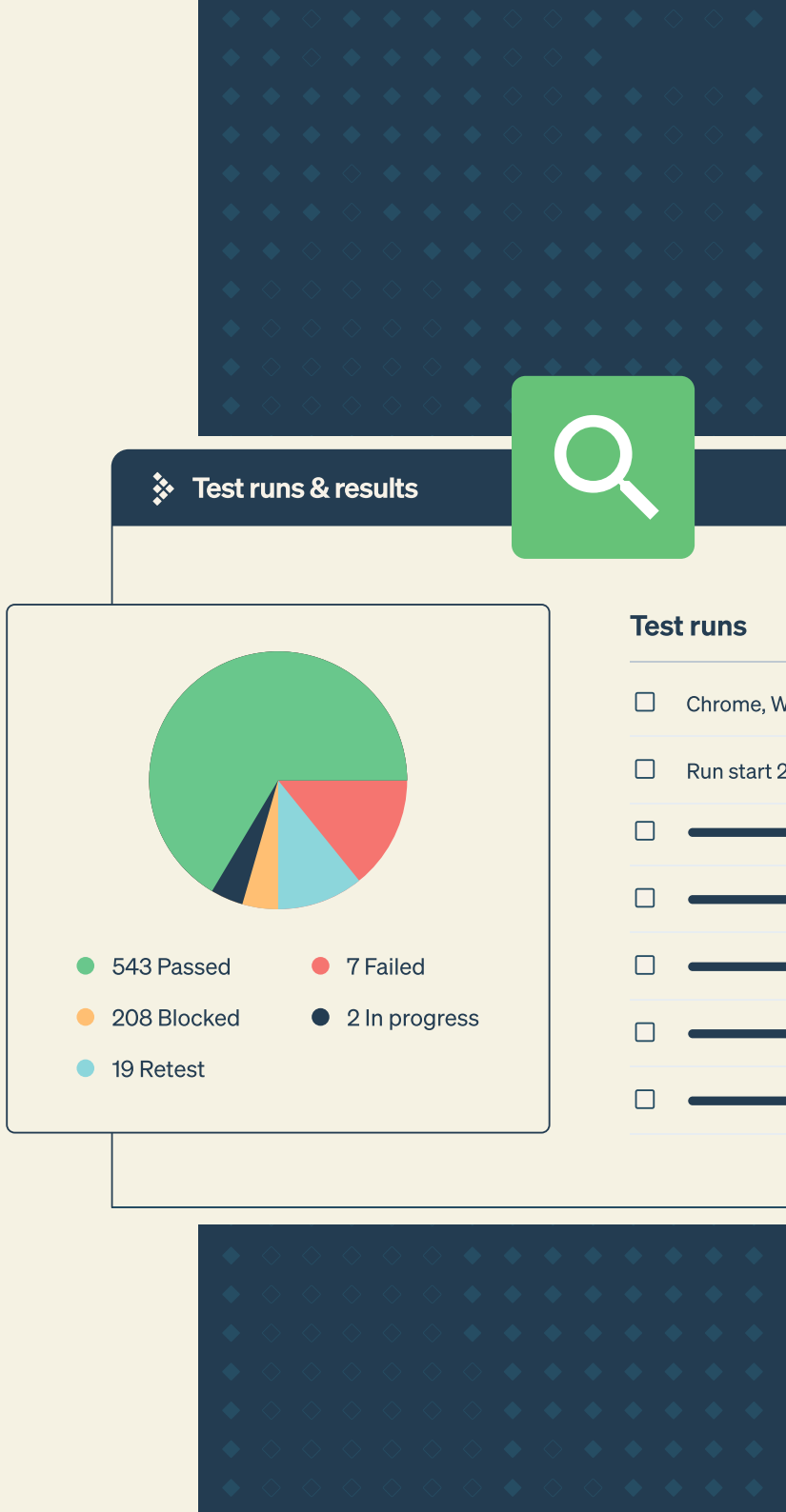
- For example, if one team member thinks a task is “done” when development is done, but you expect “done” to also encompass testing and documentation, you can easily see where a sprint might fall behind.

- Likewise, it is critical to define the criteria needed before a user story is ready for development—otherwise your team may waste valuable sprint time in developing and testing poorly-defined features.

As part of testing’s quality gates, QA should define a certain percentage of test coverage that needs to be achieved before a sprint is considered “done.”

- QA can accomplish this by advocating for the development of unit tests, which have a code coverage percentage associated

- QA can also refer to static code analysis, which should have a positive score



Make Your Testing Process Transparent and Visible to the Whole Team

01

CRAWL

Your team has defined “done” and “ready” and have a common understanding of each sprint’s goals. You are assessing the integrations, functions, and capabilities that you’d need in a test management platform and are considering your options.

02

WALK

You have selected and implemented a test management platform and introduced it to your team. You have set the expectations that this is your team’s single source of truth for all testing data. You have integrated it with your major systems, such as test automation and project management platforms.

03

RUN

Your team is comfortable with your test management system and able to confidently find the information they need. It is frequently referred to in meetings and stand-ups. You are utilizing its reporting capabilities to provide visibility to project stakeholders and executives.

Section 04

Make Your Automation Agile

Make Your Automation Agile

What is the role of automation in agile testing?

- Allows your team to focus on testing more valuable, riskier areas of your application without sacrificing test coverage
- Completes repetitive, static regression testing faster, helping to ease the burden of testing previous releases
- Reduces strain on human testing resources, which are often stretched thin on agile teams

Automation is integral to an efficient, scalable agile testing program—but only if you automate intelligently.

Make Your Automation Agile

The first question you need to ask yourself is "which tests should I automate?"

It may be tempting to answer "everything," but testing fewer things more effectively is a much more agile approach than just automating the entirety of your existing test suite.

To automate or not to automate. That is the question!

Download our interactive automation scoring model to help you prioritize what to automate next.

Download Now

Which Tests Should You Automate?

These questions can help you narrow it down:

- Is the test going to be repeated?
- Is it a high-priority feature?
- Do you need to run the test with multiple datasets or paths?
- Is it a Regression or Smoke Test?
- Does this automation lie within the feasibility of your chosen test automation tool?
- Is the area of your app that this is testing prone to change?
- Can these tests be executed in parallel, or only in sequential order?

Make Your Automation Agile

Next, you need to answer “when do we automate tests during our sprints?”

There are two approaches here:

Executing development and test automation activities simultaneously within the same sprint

- Gets tests automated faster, but need full-team dedication to execute efficiently
- Can get difficult when features are developed late in the sprint

Alternating efforts, so features developed in one sprint are automated in the next one

- Ensures steady progress with both development and test automation even if you don’t have a large team able to dedicate their time to both
- However, automation efforts will always be on a one-sprint delay and manual testing must be performed in the meantime

Want to learn more about developing a test automation program that delivers consistent results? Download our free ebook, **Trigger Your Test Automation Strategy: The QA Leader’s Guide to Effective Test Automation**

Download Now

Make Your Automation Agile

01

CRAWL

Your team is establishing a process for when tests will be automated during each sprint and what that workflow looks like. You are considering your options for a test automation framework if you don't have one already.

02

WALK

Your team has an automation framework and approach in place, assigned clear responsibilities for who will maintain your automated tests, and has set an ambitious but achievable goal for the number of regression tests you plan to automate each sprint.

03

RUN

Your functional regression testing is fully automated and you are beginning to explore automating other types of testing (like API or UI-testing), and your team is in a comfortable flow in regards to developing new automated tests each sprint. These efforts have reduced strain on your testers, who now have more time and resources available to do more valuable kinds of manual (or "human-centric") tests like usability, exploratory, or user acceptance testing.

Section 05

Measure What Matters

Measure What Matters

The core of Agile is the practice of continuous improvement: both within the product your team is developing, and within your own processes.

Quickly and accurately measuring impactful QA and development team metrics is vital in the fast-pacing agile testing cycle—without measuring anything, teams don't have a baseline to improve from.

However, not every agile QA metric is going to be a valuable one to measure.

Measure What Matters

Metrics That Matter:

- **User sentiment**
  - How happy are users with your latest release?
  - What feedback do they have?
- **Defects found in production**
  - AKA “defect escapes” or “defect leaks”
  - The number of customer or support team documented defects
- **Test case coverage**
  - How many of your requirements, stories, and acceptance criteria are covered in one or more test cases
  - Can be easily tracked with a test management tool
- **Defects across sprints**
  - Defects that “slip” from one sprint to the next due to time and resource constraints
  - The more a defect slips, the more likely the defect ends up in the final production release
- **Committed vs Delivered Stories**
  - The number of stories that were committed to a sprint vs. how many were actually delivered
  - Measuring the number and type of stories that slip or slide from sprint to sprint indicates a planning issue and a story development problem

Metrics That Don’t Matter:

- **Defect counts**
  - Can pit team members against each other, especially if defects are tracked by tester
  - Works against the Whole Team approach that is vital for Agile success
- **Hours**
  - Agile teams work best when teams are self-driven, motivated and then trusted to deliver
  - Tracking hours or requiring testers to log their activities erodes that trust and creates unnecessary work during already tight sprints
- **Lines of Code or Defect Fixes per Developer**
  - Encourages team members to write unnecessary code and/or select simpler defects to fix in order to adjust numbers in their favor
  - Can be looked at holistically to measure project progress, but are not useful metrics to measure individual performance

Adapt Your Test Approach to an Agile Environment

A comprehensive test management platform–such as TestRail—makes it easy to consolidate all of your testing data in one place and easily pull and distribute reports.

01

CRAWL

Your team is defining which agile QA metrics matter most to your testing efforts, and are establishing processes for collecting this data.

02

WALK

Your agile QA metrics are accessible in an easily-accessed repository with reporting capabilities. You have collected enough data to establish baselines, and start to compare current progress to historical progress.

03

RUN

Your team frequently refers to your agile QA metrics and uses them as a basis for continuous improvement. The data is regularly analyzed and negative trends are identified and corrected before they become a problem that impacts multiple sprints.

Section 06

Find the Balance of ‘Just Enough’ Documentation

Find the Balance of 'Just Enough' Documentation

The Agile Manifesto emphasizes working software over comprehensive documentation—a lot of teams interpret this wrong and forego documentation altogether in the name of saving time.

The key to documentation in agile testing is finding ways to do just enough documentation — no more, no less.

Test documentation can be time consuming, but it is necessary. However, agile teams **don't need as comprehensive documentation** as traditional waterfall testing teams, as long as your team is truly agile.

- When everyone knows everyone else's sprint tasks and the team has regular standups, you don't need as much documentation as you would on a large waterfall team where testers unfamiliar with your work may need to take it over.

Find the Balance of 'Just Enough' Documentation

Any agile document you create should have **just enough detail to serve its purpose** and communicate needed info—focus on reducing waste and saying more with less.

Some commonly used documents in agile testing:

- **Test strategy for the overall system**
- **Test plan for each sprint**
- **Test cases**
- **Test ideas and test logs for exploratory testing**
- **Checklists for regression testing**

A test management platform can make it faster and easier to record, store, and reuse test documentation.

One agile test case approach is to document the tests as **high-level scenarios,** with a one line description of the test and a column for details like test data or the expected outcome.

- When executing these tests, the tester may then add relevant information for future regression cycles, as well as document test results and any defects.

Adapt Your Test Approach to an Agile Environment

01

CRAWL

You are identifying the documentation that is truly needed for your agile testing process, and defining exactly what that documentation needs to include. You are looking into methods that allow you to easily reuse frequently needed items such as test cases.

02

WALK

Your team has been introduced to your agile test documentation approach and is getting comfortable with including it in their workflows. They have tools available to them to allow them to reuse documentation wherever possible.

03

RUN

Your team is fully confident in producing the appropriate amount of documentation and reliably does so each sprint. Their documentation processes are streamlined, templatized, and reusable wherever possible.

Section 07

Define Processes for Continuous Improvement

Define Processes for Continuous Improvement

Now that you're running with Agile concepts, it's important not to slow down now.
Like all things Agile, your team's success will be a continuous process.

Execute Root Cause Analysis (RCA) on escaped defects that are found in production

- Establishes a "blameless" team culture that is focused on improvement NOT criticism
- Promotes accountability and ownership for quality across all roles within the team

Track and maintain a backlog of quality/testing "tech debt"

- Actively tracking and managing quality/testing technical debt gives a more accurate picture of potential quality shortcomings to the product owner
- Allows the team to more easily determine what is highest priority testing technical debt to work on
- Facilitates a location for managing action items that come out of RCAs

Quality and testing should be front-and-center topics when doing team sprint retrospectives

- Have honest discussions about the team "temperature" regarding the level of quality the team is producing
- Identify anti-patterns and areas that require improvement

TestRail

Orchestrate Testing. Elevate Quality.

Transform your testing into a centralized, scalable process with the Quality OS by TestRail. It's time to transform your dreams of faster, frictionless releases into reality.

Try a 30-day free trial or register for a demo today:

Try TestRail

Get a Demo