



# Agile Software Testing

Ten Tips for Understanding Agile Development  
and Making it Work for You

---

**White Paper**  
August 2012

In-The-Wild Testing for Functional + Security + Load + Localization + Usability



## White Paper

# Agile Software Testing

## Ten Tips for Launching and Testing High Quality Apps in an Agile Environment

### Table of Contents

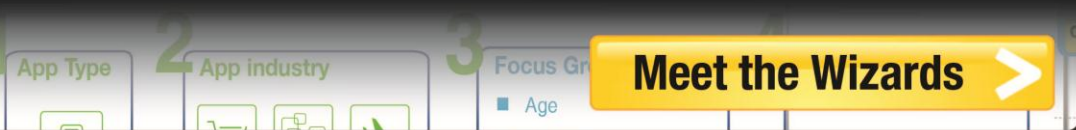
• <b>Introduction</b>	
- Agile Overview.....	3
- The Agile Manifesto.....	3
- The Problem with Waterfall.....	3
- The Emergence of Agile.....	4
- Implementing Agile.....	4
• <b>Agile Testing Tips</b> .....	4
1. Understand the Role of Testing.....	4
2. Define and Understand.....	5
3. Heads-Up: Switching is Hard.....	6
4. Common Concerns .....	7
5. Making Agile Work for You.....	8
6. Capturing Meaningful Data.....	9
7. Scrums .....	10
8. Being Agile without Burnout.....	10
9. The Big Picture.....	11
10. Agile Isn't for Everyone.....	12
• <b>About uTest</b> .....	13

**“Simplicity is the ultimate sophistication.”**

- Leonardo da Vinci



**Let our exclusive Test Planning Wizards help you customize and refine your testing needs.**



## Introduction

### Agile Overview

Believe it or not, the Agile methodology has been around for more than a decade. In that time it has been talked about, taught, tweaked, adapted, reviewed, rejected, heralded and ultimately worked its way deep into the software development culture. But still, some people scratch their heads and wonder what Agile is exactly (let alone how it applies to testing). So let's start from the beginning with a quick Agile review. If you've heard this story before feel free to skip ahead to the next section: [Agile Testing Tips](#).

### The Agile Manifesto

In 2001, a small group of CTOs, academics and thought leaders published the well-known Agile Manifesto. Here is a summary of the document's fundamental principles:

*We are uncovering better ways of developing software by doing it and helping others do it. Through this work we have come to value:*

<i>Individuals and interactions</i>	<i>over</i>	<i>processes and tools</i>
<i>Working software</i>	<i>over</i>	<i>comprehensive documentation</i>
<i>Customer collaboration</i>	<i>over</i>	<i>contract negotiation</i>
<i>Responding to change</i>	<i>over</i>	<i>following a plan</i>

*That is, while there is value in the items on the right, we value the items on the left more.*

There have been multiple "spin-offs" of Agile since this document was first published, including Extreme Programming, SCRUM, DSDM, Adaptive Software Development, Crystal, Feature-Driven Development, Pragmatic Programming and Agile-Fall, among others. Some of these have come and gone while others have stuck around, and though they differ in subtle ways, they all hold the same foundational values described above.

### The Problem with Waterfall (it's not agile!)

Waterfall development is characterized by well-defined phases, each having a thorough review and authorization process that should be completed before proceeding. On the surface, this process is not inherently bad. In fact, it's quite logical: First, figure out everything you need. Then design it. Then code it. Then test it. And repeat.

Problems arise when the project requires adjustments and modifications that were not anticipated in the early stages. Reality ruins the plan. So for projects that *can* be spec'd out with precision ahead of time, waterfall may be the proper choice. Unfortunately, experience has shown that a calm, unchanging set of requirements is a rarity.

Looking back, it's no wonder why these projects became the source of such frustration; basically, we delivered software that lacked the most important features! At the same time, we were up to our necks in fancy-looking documents – MRD, SRS, Architecture diagrams, APIs and Test Plans – that didn't provide any real value to customers.

## The Emergence of Agile

Out of necessity, development teams began shifting to frequent releases. At first, these iterative methods went in small cycles, but remained with a fixed schedule ahead of time (i.e. fixed budget, strict plan and predetermined feature sets). So instead of one major release, several minor releases would be scheduled.

This concept would be taken a step further: Iterative releases, with no fixed plan! Here, each release adds a bit of value, allowing for better decisions as to what features would be included next. Development now works in small cycles – planning and designing only far enough ahead to keep the flow working. With this increased flexibility, the burden of excessive formal documentation was greatly alleviated. Now testing is incorporated into each step of the process, not just at the end of each major release.

***“Agile methods derive much of their agility by relying on the tacit knowledge embodied in the team, rather than writing the knowledge down in plans.”***

- Barry Boehm  
Software Engineering Guru

## Implementing Agile

That's how Agile came about and why it has found a foothold in so many organizations. But this whitepaper is not an endorsement of one methodology over another. Instead, this piece is designed to help you further understand how Agile development, and specifically testing, fit into an organization. It will take a deeper look at the steps necessary to make Agile successful, some common concerns transitioning companies face and a broad overview to help you decide if Agile is the right fit for your organization.

## Agile Testing Tips

Whether you are already operating in an Agile environment, or just want to adopt more real-time, real-world testing processes, here are some things to consider to ensure you get the most out of Agile.

### 1. Understand the Role of Testing

People often think the goal of testing should be to simply find all the bugs. Well, that's not entirely realistic. Actually, it's impossible! Despite the title of “quality assurance” testers can never truly assure quality, nor can they be expected to find every bug. The real goal of testing should be to *improve the software*. This is a widely held view of testing in the Agile world.

Despite this, upper management still tends to believe that testing can uncover all the bugs or prove whether or not the software will function as intended. They wrongly equate testing with validation. To understand the true role of testing, one needs to understand that it is a scientific method – a continuous search for information. So instead of thinking of tests in terms of ‘pass’ or ‘fail,’ try to think of whether or not they provide valuable information. In an Agile environment, testers are called upon to provide this valuable information throughout the development cycle, as opposed to issuing a ‘pass’ or ‘fail’ grade at the end.

***“Testing is not a phase on Agile teams, testing is a way of life. Agile teams test continuously. It’s the only way to ensure that the features implemented during a given iteration or sprint are actually done.”***

- Elisabeth Hendrickson  
Founder, Quality Tree Software

If Agile is to succeed, **testing** must become a central pillar of the development process. Incorporating testing at the end of a major project will only lead to show-stopping bugs and major timeline setbacks. If you truly want to ensure that the product you’re creating is high quality, why not include quality assurance personnel from the very beginning where they can spot issues early on and prevent major problems down the line. If you don’t fully understand the importance and role of testing, you will be missing out on a major component of Agile.

## 2. Define and Understand

Though Waterfall is the methodology more commonly associated with thorough planning and project comprehension, understanding and defining the work ahead and project goals is no less important to Agile. In fact, the iterative sprint style of Agile requires a deeper level of understanding since a sprint is not complete if any portion remains undone. Here are a few key areas that require specific understanding:

### Testing Matrix

Defining the coverage requirements is an essential part of the specifications process, and thus can be dealt with at an early stage. The size of the coverage matrix has a significant impact on staffing needs, time requirement and QA costs. Because of this fact, defining it clearly and early helps QA teams understand what needs to be done and begin testing earlier. It also helps management allocate resources and determine what can be done in-house versus what should be outsourced for greater coverage. This applies to:

- Operating systems
- Browsers
- Plug-ins, anti-virus programs and firewalls
- Geographic locations
- Languages
- And in the case of mobile apps, manufacturers, devices and wireless carriers

Not understanding your target demographic and the real-world conditions they'll be using an application under will lead to wasting time testing on hardware/software combinations that aren't popular or missing combinations that will leave a giant gap.

### Goals

Defining the goals of a sprint at the beginning gives teams a clear idea of what they are working toward. Well understood goals allow teams to know when a sprint is done (another extremely important aspect of Agile development) and protects against feature creep. Segmenting a team and making testers wait until the end is not an efficient way to reach a goal. Remember, optimization improves quality. Testers know how to find errors. They also know that errors don't just appear at the end, they appear throughout the process. Finding errors in logic or in usability from the very beginning is a must for a successful, productive, efficient sprint.

### Defining Done

Gartner identified this as a key component to successful Agile development. Similar to understanding a sprint's goals, defining when a sprint is completely, truly, 100%, totally done means that everyone on the team is on the same page. Agile relies on how well a small group of people can achieve a common goal. If everyone on the team understands what needs to be complete – from soup to nuts – it will make working toward that goal easier. Team members will no longer make up arbitrary estimates ("I'm 82% done") just to impress, or declare themselves done and ready to move on to the next phase unless all team members are also done. This is the advantage of daily scrums.

Fixing broken windows is a perfect metaphor for this issue. In the 1980s, an experiment was run to measure the impact that broken windows have on crime in urban neighborhoods. While broken windows certainly weren't a *cause* of crime, it was shown that just fixing broken windows in a neighborhood helped decrease the crime rate. The reason for this is that if you give the appearance that you don't care, then others won't care either.

It's the same with your testing efforts. If you continually say "Ignore this bug, it's a known issue" or "Ignore that test, it's not yet relevant," your testing team will begin to listen to you. They'll be more willing to ignore bugs and they'll miss the ones that you don't want ignored. Agile development is the practice of completing small, set portions of a product and launching them independently. If you are ignoring bugs and moving on you are not completing sprints. This defeats the purpose of Agile development and takes away many of its benefits.

### 3. Heads-Up: Switching is Hard

Unless you are a brand new company, adopting Agile will require change. Do not make the mistake that this will be a quick, easy transition. One of the biggest complaints from companies implementing Agile is that upper management expects the methodology to work perfectly from day one. These are unreasonable expectations. Development and

QA departments will need time to learn, understand and work the quirks out of this brand new way of doing their jobs.

If you do decide to switch to Agile, be sure the Agile teams – and the executive level overseeing their work – fully embraces the practice. This will help you avoid the pitfalls of “Fake Agile.” Fake Agile is a term coined by Software Tree Company founder Elisabeth Hendrickson, “it removes all the external controls that made traditional practices work without instilling any of the team-centered disciplines that make agile work. The end result is usually worse quality, slower and an organization that blames ‘agile’ for the mess.” (Note: “Fake Agile” should not be confused with Agile-Fall or other hybrid practices.)

***“Out of 200 participants, 64 percent said that switching to Agile Development was harder than it initially seemed.”***

- Market Snapshot: Agile Realities  
Voke

Fully embracing Agile may require that some potentially separate departments be combined, or, at the very least, members from all participating departments should be included on the Agile team. If a development team adopts Agile but does not include the QA or operations departments, the benefits of Agile are wasted. All departments that have a direct hand in planning, creating and testing a product will need to practice Agile together if they have any hope of completing the project in quick, independently complete sprints.

#### 4. Common Concerns

Making such a radical shift can understandably be nerve-wrecking and management is sure to have some questions and concerns. Here are a few of the more common concerns about Agile development, and why you shouldn't be worried.

##### Lack of Planning Time

The better part of Waterfall development is spent planning the steps to come. Switching to Agile can leave some teams feeling crunched for time or lost without this predetermined roadmap. But the need to define and understand the testing matrix, goals and “done” require planning.

Planning for Agile development is simply segmented. A team picks a feature to work on and plans for that feature and that feature alone. As the teams move through the project as a whole, they may identify aspects that were inadvertently left out of planning in the first few sprints. With Agile this is a much smaller problem. The nature of Agile means missteps affect only small portions of the project and corrections can be made quickly and easier in future sprints. If something is missed during the planning phase of Waterfall development it is much more disruptive when discovered down the road.

##### Lack of Documentation

A key component of Agile is doing away with unnecessary, clunky documentation. This, however, does not mean getting rid of *all* documentation. Many teams use index cards



or a gridded whiteboard to represent sprint requirements, goals, step order and to show what each team member is working on at a given time. Documentation does not go away, it just becomes leaner.

Connected to the idea of defining done, a sprint is not truly complete until all necessary product documentation is complete. This traditional documentation isn't missing, it's just completed in parts rather than at the end of the entire project.

### Daily Meeting Hassle

While the prospect of a daily meeting may seem like a giant waste of time, the purpose of Scrum is to be a quick check-in and organization meeting. We'll go more in depth on successful Scrum practices later in the whitepaper, but if you honor the purpose of Scrum you will find it is more useful than hassle.

## 5. Making Agile Work for You

Several off-shoots of Agile have risen to the surface in recent years. These off-shoots – like Agile-Fall or Scrumfall – are the byproducts of organizations attempting to adopt Agile practices and ultimately settling on a hybrid practice that works best for them. These hybrids aren't bad, they are just another option for dev and QA teams to get the job done. Some companies decide they'd like to be more Agile but need more lead time before projects or more documentation during sprints. The important thing is to find a method that works for your specific situation. As long as the main goal of producing better products quicker and with fewer resources is maintained, the spirit of Agile is intact.

If you are unsure if Agile will work for your company, or are just weary of a major simultaneous shift, consider designated some less critical projects as Agile test runs. This will give you an idea of how Agile development works within your specific organization and will help work out the kinks before the method is used on high-level projects.

***“There’s no shame in [hybrid methods] if that’s what works, and when you’re going through a transition from Waterfall to Agile, that may be the best thing as opposed to a sudden lever-pull one day where you show up and your desk is next to someone else with no walls and there’s a stack of sticky notes and markers on your chair with an email to report to your first standup in 30 minutes.”***

- Jon Bach  
Director of Live Site Quality, eBay

Whether you embrace Agile whole-heartedly or adopt certain aspects of the method, it is extremely important that you identify a product owner. This is the person that understands the project's big picture, knows how sprints fit together and acts as the go between for the Agile teams and upper management. If the product owner does not fully understand the product or how Agile development works they will not be efficient. Having a dedicated individual in this position and giving them the access, resources and decision-making power necessary to get the job done is crucial to successful Agile



development and a successful product. Take the time to identify this person within your organization and ensure they understand the project's goals. Then trust them.

## 6. Capturing Meaningful Data

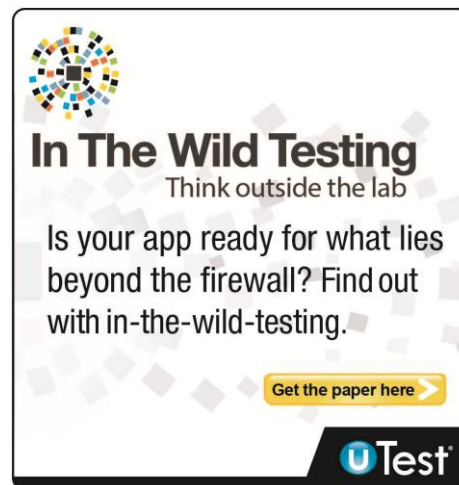
Testing managers often focus on information that is of little use to anyone but themselves – keep this in mind when collecting data. Compiling detailed reports that show the number of test cases written (i.e. how many have been run, failure rate, severity levels, etc.) is certainly valid data to the testing manager, but who else in your organization will care? Would the business development team, for instance, have any practical use for this? Remember, Agile is all about minimizing paperwork and only spending time on documentation – including data – that is absolutely necessary. As testing expert Michael Bolton said:

“When you enslave numbers, they eventually rise up, revolt and enslave *you*. These organizations spend so much time collecting the data and talking about it and justifying it and trying to duck blame that they don't seem to have time to do anything about the actual problems, which generally fall into two categories. One: the organizations are trying to do more work than they can handle with the approaches they're using. Two: they're not listening to people that matter – neither to their customers, nor to their own front-line staff, many of whom are closest to the customers.

When your car is about to go off a cliff, it's a weird time to be thinking about gas mileage and drag coefficients; better to take the right control action – look out the window and steer or use the brake until you're back on course.”

Focus on data that will be helpful to several departments and will further the efforts of the Agile teams. One type of data that would be of value is direct business impact statistics: Sales missed due to bugs, customer renewal rate and relative cost of defect. This type of information is like gold to decision-makers and will help the cause of development and test teams.

Using crowdsourced or beta testing will also provide useful data in the form of end-user feedback. Both crowdsourced and beta testing gives you access to testers who mimic the demographics of your target market and their feedback can affirm (or refute) any preconceptions you might have regarding your software. You will get to see your product used under real-world conditions, something that often does not occur in many QA labs. Better yet, this information can be shared with other key decision makers in your organization, notably the sales and marketing



departments, who'll often spend great amounts of money on surveys and research that are not very Agile.

## 7. Scrums

Far from an Agile prerequisite, daily Scrum meetings have proven to be extremely beneficial for teams in the long run, as they help departments remain focused, energized and coordinated. Daily Scrums keep team members responsible to each other. Each team member knows what the others are working on and where their piece fits into the puzzle (or conversely, where their work is making others wait). Transparency and connectivity are staples of an efficient team. Writes the Scrum Alliance:

In most companies, development is slowed down by issues identified as impediments during the daily meetings or planning and review meetings. With Scrum, these impediments are prioritized and systematically removed, further increasing productivity and quality. Well-run Scrums achieve the Toyota effect: four times industry average productivity and twelve times better quality.


Scrum removes management pressure from teams. Teams are allowed to select their own work, and then self-organize through close communication and mutual agreement within the team on how best to accomplish the work. In a successful Scrum, this autonomy can significantly improve the quality of life for developers and enhance employee retention for managers.

Too often, however, daily Scrum meetings become weekly meetings, which in turn become monthly meetings. Pretty soon, you no longer recognize Bill from development. So set a time that works for everyone, re-introduce yourself to Bill, and do your best to stick to the schedule. It also helps to appoint a Scrum Master who will lead the meetings and ensure the Scrum schedule is adhered to.

## 8. Being Agile Without Burnout

The short release cycles of Agile development can often place QA teams under tight time constraints. Luckily, there are natural gaps in work schedules. A good way to stay on schedule while giving teams a much needed break is to leverage outside help whenever possible – particularly during natural breaks. The weekend is almost always a period of low development and testing activity, and this is a good thing – people need their rest to maintain productivity. However, this two day gap need not go to waste.

By adding a crowdsourced community to your testing process, you can deliver software releases for testing each Friday and receive the results by



**uTest vs. Outsourcing**

Lower costs, better coverage, faster cycles.

Discover the benefits of crowdsourcing now.

[Get the paper here >](#)

Monday morning, thus compressing the development cycle without adding pressure to in-house staff. Crowdsourced testing is a better option than traditional outsourced testing for these situations because crowdsourced models tend to offer more flexibility and quicker ramp-ups. Where an outsourced team may take weeks to schedule and prepare a test, a crowdsourced team can often be put together in a matter of days or even hours. Crowdsourcing also gives you the chance to cover your entire testing matrix – no matter what device, browser, platform version, carrier or location – which would be nearly impossible in house. By adding crowdsourced testers, Agile teams of all sizes are able to target very specific users for specific assignments within their tight sprint schedules.

## 9. The Big Picture

Focusing so much on individual sprints makes it easy to forget about the big picture final product. Yes, the product of each sprint is intended to be independently complete and functional, and often released on its own. But how do all the pieces work together? When more than one piece of the puzzle comes together it's important to remember routine responsibilities, like regression testing, load testing and overall security testing.

### Regression Testing

At the end of each sprint it is important to test that sprint's product against pre-existing pieces. While the sprint may have been thoroughly testing on its own, it is possible for this new release to adversely affect a previously completed component. Never forget that in the end, all sprints are working toward one common, functional goal – a successful, multi-faceted product.

### Load Testing

At what point will your application's performance begin to degrade? How many concurrent users can it support? How do the JavaScript-based components of your application behave with 50, 100 or 1000 active users? Where are the bottlenecks between your code base, database, CDN and load balancers? Without answers to these questions, your testing could be all for naught. Load testing on the completed end product is more important than load testing each sprint. The strength of each brick can be verified, so to speak, but that doesn't ensure the structural integrity of the wall. If your software can't hold up under stress, it doesn't matter how agile your methods are.

### Security Testing

Each sprint should undergo its own round of security testing to catch any major holes in that particular component. But it is also important to do an overall security test and evaluation at the end of the project. You want to make sure hackers can't sneak in through the gaps after the software has been stitched together. You also want to be sure that data isn't leaking from some unaccounted for area that didn't make it into any of the sprints. Users are becoming more and more aware of software security, and they're becoming correspondingly intolerant of security snafus. Be particularly careful that adequate security testing is done in this new world of less documentation.

## 10. Agile Isn't for Everyone

There is nothing wrong with other methodologies. Do not switch to Agile simply to switch. If you think your team, department and, ultimately, company will benefit from Agile then give it a shot. But if everything is working perfectly smoothly, teams are happy and successful products are being launch on time and budget then Agile may be an unnecessary, burdensome change. Similarly, if thorough documentation and in depth accountability is necessary for your industry, Agile could be down-right detrimental.

If your organization uses traditional waterfall methodologies, you may still be able benefit from some Agile concepts to improve quality. If you try Agile and it simply isn't working, revert to the old method. Remember, it's about embracing what works for your company, not adhering to a development methodology for the name alone.

***“Adopting Agile won't fix a dysfunctional team and it won't help an organization to learn to accept its limitations and work within them.”***

- Noah Sussman  
Test Architect, Etsy

If your company is developing quality software; if the development is sustainable and the business is being adequately served by that software, there's really no need to press the issue. Agile is no panacea for problems related to development, process and management.

So before you switch to Agile, take a moment to review your current practices, flag the major issues you're hoping to correct, study and fully understand how Agile development works and objectively evaluate if Agile will be able to help your company. Take your time switching and be sure all the key players are on board. Agile development can work wonders for companies hoping to produce higher quality software faster and more efficiently, but only if they do it right.

Good luck!



## Ask an Expert

**Have testing questions?**  
**Let one of our testing experts guide you.**

[Ask an Expert >](#)

### About uTest

uTest provides in-the-wild testing services that span the entire software development lifecycle – including functional, security, load, localization and usability testing. The company's community of 70,000+ professional testers from 190 countries put web, mobile and desktop applications through their paces by testing on real devices under real-world conditions.

Thousands of companies – from startups to industry-leading brands – rely on uTest as a critical component of their testing processes for fast, reliable, and cost-effective testing results.

More info is available at [www.utest.com](http://www.utest.com) or [blog.utest.com](http://blog.utest.com), or you can watch a brief online demo at [www.utest.com/demo](http://www.utest.com/demo).



**uTest, Inc.**  
153 Cordaville Road  
Southborough, MA 01772  
**p:** 1.800.445.3914  
**e:** [info@utest.com](mailto:info@utest.com)  
**w:** [www.utest.com](http://www.utest.com)