uTest®
Where the best test

The Future of Beta Testing

6 Tips for Better Beta Testing

White Paper
September 2012

White Paper

The Future of Beta Testing

6 Tips for Better Beta Testing

Table of Contents

> **"Educators assess student's knowledge, retention, and synthesis of the material via tests. The results of the tests provide information to the teachers/educational institutions to help them make decisions such as 'Is the student prepared for the next phase of learning?,' 'Should more time be spent on this topic?,' or 'Has the student achieved sufficient learning for us to feel proud?' ... Testing software is the same."**
>
> \- Scott Barber
> Chief Technologist, PerfectTestPlus

You might also like …

Introduction

Your application has been developed, tested and refined by your internal teams for what seems like an eternity. You've finally released it to the public, but you're not *entirely* done with testing, bug fixes or even development of new features. There's a name for this awkward state between development and launch: **it's called beta testing**.

For years, companies have used this phase as a "dry run" to solicit user feedback and learn more about how their application is working (or not working) in the hands of real end-users. And for years, companies have been frustrated by the results, encountering problems such as:

> **"A principle objective of software testing is to give confidence in the software."**
>
> - P. D. Coward

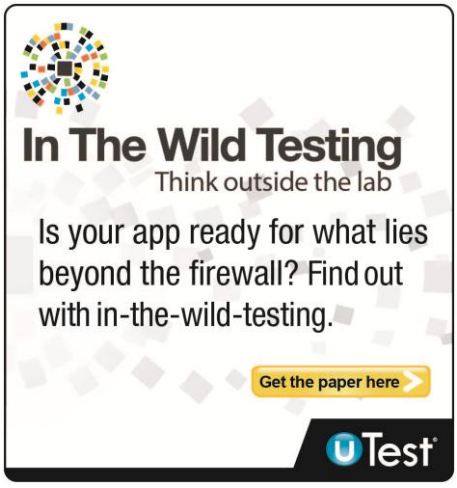
- A lack of actionable and accurate feedback
- Inconsistent participation (too much or too little)
- Coordination and scheduling problems
- Negative press and user reviews
- Poor processes for collecting and analyzing feedback

Despite these problems, companies insist on following the same model for beta testing and expect to get different results. Without a major change in their approach, it's not going to happen.

In this brief whitepaper, we'll show you how to avoid the common pitfalls of beta testing and get more from your early release and public-focused testing efforts.

What is Beta Testing?

The meaning of the term "beta testing" is highly subjective. For some companies, a beta product could mean that it has been tested, refined and rolled out to millions of users (think Gmail). For others, it means that an application has been released to a dozen testers with a strict gag order. Typically, however, when software is referred to as a "beta test version," it is being released to an audience outside of the development and QA teams – typically a segment of the intended end user audience. The hope is that by the time software reaches beta testing, most of the major defects have been removed by the developers and in-house QA teams.



Beta test participants are *expected* to provide the development team with feedback and bugs that can be addressed before a full launch. The key word here is expected. As

many companies have come to realize, beta testers are NOT professional testers. Yes, they will almost certainly be able to provide you with a higher quantity of feedback, but the quality is likely to be incomplete, inconsistent and certainly not actionable.

The key point is this: Beta testing extends QA, but it can never *replace* it. We'll touch on this theme throughout the whitepaper.

A Quick Look Back: The History of Beta Testing

The first instance of "beta testing" in the modern sense can be traced to IBM in the 1950s. During IBM's hardware development they referred to any testing prior to product announcement and release as Beta. In the 1960s IBM dropped the term, but by then it had gained substantial attention from the outside world. Moving forward, beta testing flourished with the growth of the software industry.

The Evolution

Of all the companies which have played a role in the evolution of beta testing, Google has been the biggest game changer. A prime example is Gmail, Google's email client. It began running in March 2004 with a focused beta test. Google invited a carefully selected focus group of 1,000 people to use Gmail beta and invite their friends and family to join them. One month later, Google opened Gmail to the public, still in Beta. Very minor security flaws and issues were discovered and patched during 2004-2005. The product was essentially in a finished, usable condition. However, Gmail did not officially exit beta until July 2009 making for a five-year "beta phase" where the product was completely usable and fully publically accessible.

> **"Often testing went on for years before a product was released. By the late 90's and early 2000's testing practices had to evolve as smaller companies with lighter and more flexible delivery approaches challenged this paradigm."**
>
> - Anne-Marie Charrett

The reason Google's products (like Gmail) are more often than not fully functional while in beta stems from the fact that the company does two other major forms of testing prior to beta testing. Google first performs up-front testing where internal users test in the lab. Next comes crowdsourcing, where they test outside the lab based on user scenarios. This is all before the software is released into beta testing.

After the release of several Google launches, the definition of beta became blurry. People saw a "beta version" as a finished product, and few remembered it as a test.

Conversely, in October 2011 Apple released a beta version of Siri – Apple's voice-activated virtual assistant application. Releasing features for public beta tests is uncommon, if not unheard of, for Apple. When Siri was released, the application contained several bugs and inaccuracies. Despite the fact that Siri was in beta form (and

labeled as such), users threatened and criticized Apple for producing a poor product. Apple took a good amount of heat for releasing Siri, but their goal was to give it as much exposure and real world testing as possible. Nevertheless, customers expected the beta version of Siri to be fully functional and to "just work." This expectation was fueled in equal parts by Apple's reputation for releasing near-perfect products (without beta testing) and Google's redefining of the term "beta" that desensitized the public to the true meaning of the word.

The point of these examples is to illustrate just how much the definition has varied. Nevertheless, many of the day-to-day applications we use have been beta tested in some form. From an email client, to a favorite website or app, it's likely these tools were at some point in a beta release.

Types of Beta Testing

There are two major methods of performing beta testing, known as public and focused beta testing. They are very different from one another, and while they can be effective, they both have their pitfalls. Let's take a close look.

Public Beta Testing – The Good

A public beta test is openly distributed online, and data is gathered from anyone who responds. Anyone may access the software without limit. Public beta tests can be used to raise awareness for a near-release, and/or to find issues and make product improvements. Beta testing with the public can provide real world information to companies that are looking to verify their application's performance and functionality.

In many ways, public beta tests are easier to set up than focused beta tests because there is no recruitment process. Anyone is allowed to enter, often creating an enormous pool of users across the globe. This means you are testing on a wider variety of systems. This is useful for knowing how your app functions in real world scenarios (i.e. for a mobile application, does the app function on an iPhone in a rural location?)

Public beta tests are also helpful for an app development team looking to market internationally. If users in different locations participate in a public beta test, it is possible some localized issues will be reported. Every country's language, culture, currency, taxes and standards are different. Therefore, it is important to know how your application functions in different aspects of international culture.

When you involve the public in a beta test, you are often involving your customers as well. An obvious benefit of this is that companies are able to test with their key audience. By doing so, developers get a clear view into how their software will fit into their target market. In addition, if testers have already used your products, they might be more familiar with how to use the specific software features of this new release. This can generate better quality feedback. It also encourages word-of-mouth that will help your application gain traction in the market leading up to release.

Public Beta Testing – The Bad

There are some major concerns associated with public beta testing. For one, the goal of testing in general is to find any bugs, speed or performance issues before the product actually hits the hands of your customers – where it could potentially cost you money and reputation. With public beta testing, developers are putting their company's reputation at risk by exposing their customer base to an unfinished product. A poor public beta test can backfire and lead to disappointed customers not downloading the final release.

Another challenge with public beta testing is that it can be difficult to accomplish the test's goals. Not all participants will provide feedback, and the ones that do often leave a generic statement – such as "it crashed" or "not as good as X" – making it difficult for developers to take away insightful feedback. Remember, beta testers are simply members of the public. They are not professional testers so they do not know how to adequately test a program, let alone write a coherent, useful bug report. If a company gets a lot of feedback from a beta test, it can become extremely challenging to manage this data, particularly when much of the feedback is unhelpful white noise.

> **"[A tester's] relationship with development is also key. Providing the developers timely, contextual and actionable feedback on the health of the application is critical to any software release."**
>
> - Matt Evans
> QA Director, Mozilla

In extreme cases, companies have released unusable beta versions publicly. Many corporations over the years have released beta versions of their software to the public that simply do not work. Needless to say, the companies have been highly criticized for this major testing faux pas. In instances like this, companies have upset their customer base, therefore hindering their success as a business. It's important to remember that beta testing should only be conducted once a product is feature complete and has been tested (at least minimally) in-house.

Another instance of failed public beta testing is when companies use beta for the wrong reason – as a marketing awareness tactic rather than for technical reasons. Some companies use a public beta test to raise awareness about their upcoming product without being concerned about product improvements. When companies take this approach, they are essentially marketing the problems their application contains, which is every company's worst nightmare.

If the app you're going to test is meant to be kept quiet, beta testing creates a dilemma. An example of this is the integration of features between Instagram and Facebook before the acquisition announcement was made. The CTO of Facebook would by no means have risked testing this integration outside of the firewall. Opening a beta test to the public leaves no control over the news.

Focused Beta Testing – The Good

Focused beta testing is a closed, or invite-only, beta test. Typically much smaller than public beta tests, many developers use focused beta testing for two reasons: If done correctly it can provide better feedback, and companies run less risk of damaging their reputation or losing potential customers.

Like public beta tests, focused beta gives you access to a wide variety of hardware and software combinations that you may not have available in your testing lab. It also covers the need to test under real world conditions – your beta testers will be testing in the same environment they would typically use the product.

A focused beta test has the added benefit of allowing an in-house QA team to hand pick which users they allow into the project. This selectivity enables dev teams to guarantee the beta testers match the end-user demographic. By being selective and precise, teams cut down on the level of feedback and, hopefully, receive bug reports that are more in line to their goals. For example, if an app is designed for rural farmers, feedback from a tester in Paris, France is not helpful. In a public beta test there is no way to stop that tester from signing up and providing unhelpful feedback. But with a private, or focused, beta test, the Paris tester would not have been allowed into the test in the first place.

Using a smaller test group also minimizes the reach of any negative feedback. If your program performs unusually poorly, only a small number of potential customers will have experienced the faulty program. If those users ultimately decided to not purchase your finished product, you will not have inadvertently gutted your entire market or destroyed your reputation on a large scale.

Focused Beta Testing – The Bad

While a focused beta test may seem like the way to go, it does present its own challenges. Focused beta testing is a multi-step process, as opposed to public beta testing where you just release your application and wait for the feedback to come.

First, you need to get the right users to sign up, which isn't always easy. Some companies just promote the beta test thoroughly, and others offer incentives for their testers. Because focused beta tests are done with smaller groups, it is imperative you get the participants you want. Many companies ask their focused beta users to request to be a tester through an application process. In this scenario, you need to monitor the process to make sure you are getting the testers you desire – which calls for more internal man-power than low over-head public betas. Alternately, you can run a closed beta test by simply having people submit their email address and wait for an invitation. This method doesn't give you the ability to ensure all testers fit into a specific demographic, but it does let you control how many people are testing your application. Limiting the number of testers helps control the amount of feedback and the potential for negative opinions leaking into public arenas.

Even though you are using select testers, it's important to remember that, like open beta tests, you are still only using everyday end users – not professional testers. Focused beta tests help cut down the noise to signal ratio, but there is still no guarantee you will get quality feedback. The only way to ensure you get a proper bug report is to use professional testers.

Beta Testing Gone Wrong

Failed betas are very common. Often, companies who release their software into beta don't receive quality feedback, and then launch the application containing bugs beta testers missed. When a failed launch happens the company usually pulls the application back into beta testing.

A very recent example of beta testing gone wrong happened in August 2012, when a company called Goko released a HTML5 web portal for developing multiplayer games. Unfortunately, when the software was in beta, the beta users did not catch a very crucial problem that caused the portal to crash. It was specifically a load problem that may have happened because the company's pool of beta testers was too small.

The company issued an apology on their website saying, "As we moved out of beta and into the live world, we learned that we had problems with our backend and weren't able to scale fast enough to support the load. As a result, we experienced numerous problems and created a really bad experience for the fans trying to access our games or even get on our website."

They then pushed their software back into beta. The application was not ready for real-world usage, but unfortunately the beta test did not give the developers that feedback. After a failed launch, it is increasingly difficult for companies to have a successful re-launch. While the product may be functional at the second launch, companies struggle to convince customers that a platform is usable the second time around. It comes down to a trust factor between the company and the customer that a failed beta thwarts.

Tips for Better Beta Testing

There are countless ways in which beta testing can go tragically wrong. In order to avoid any major pitfalls, there are a few important tips:

1. Be Product Ready

Make sure your product is ready to be tested. Before opening a beta test, eliminate any glaring errors and put your product through the initial paces. Beta testing is not for finding obvious glitches, it is a step beyond that. If a known bug is still in the product it can lead to wasted

> **"Let me generalize this — the quality of a program extends far beyond its functionality. There is a huge gap between "works right" and "works well."**
>
> - Dr. Cem Kaner
> Author, Lawyer, Speaker & Professor

time in your beta test. One obvious bug left unfixed can result in multiple reports, potentially from everyone who gets involved in the beta. A known bug can also be camouflaging an unknown issue. If you add a feature or make a product change during the beta process, you need to restart your beta. When adjustments are made to a product, it takes longer to flesh out unexpected bugs.

2. Recruit Enough Testers

To do your testing you need to recruit enough testers, both in numbers and in demographics to accurately represent your target audience. With public beta tests this is rarely a problem, but it can be more difficult for focused tests. The minimum number of serious testers you need in a beta – meaning those who understand your software, the bugs they encounter and provide quality feedback – is usually around 100. If you have several beta managers, it would be good to get 100+ serious testers for each manager. Often, companies who perform focused beta tests create an application form for testers, requesting their contact and system information. (Note that this can require providing a privacy statement to those who apply and getting certain legal documents for the selection process.) To attract the needed testers, you may want to provide incentives for comprehensive feedback. Even if you do so, only roughly one in five people will register for the beta – therefore creativity is key in your recruitment process. Be sure your beta test pool provides sufficiently diverse coverage to asses any usage issues related to location, network, device or platform version.

3. Plan your Timing

Quality doesn't happen accidentally, it has to be carefully planned. Because of the quickly changing nature of the technology market, companies often don't leave enough time for a sufficient beta test and race to release their product. What matters more than being the first to market is being first to market with a quality, bug-free product. Be sure to give your team enough time to fix the issues the beta testers find. Beta testing takes time, an application cannot be adequately tested in just a few days, or even weeks. However, the test itself can't be too long. The longer the test, the less likely you will be able to sustain the interest of your beta testers. There ultimately needs to be a balance.

4. Make No Assumptions

If you don't receive any initial feedback, don't assume your product is ready for release. The beta test could have gone so terribly wrong that the testers simply quit. Or, maybe they just haven't found time to test yet. If you do not get any feedback after the test has been running for a while, reassess how you're interacting with testers and make sure nothing in the reporting chain is broken.

5. Generate a Plan

It is easier to follow these best practices if you generate a beta testing plan before you begin. Define the beta's goals, activities, target participants and desired outcome. Stick to your goals and continue to follow through after the beta testing phase ends. Too often developers don't do anything with the data they collect from a beta test. If the feedback is of good quality, the suggestions could further the success of your product. When

multiple beta testers report an issue, it is all but guaranteed that your customers will have the same problem as well.

Take your master plan and create a tester plan that you communicate to your beta users – particularly if it's a focused test. The plan you share with the testers should contain an overview of the information that they will need, including a summary of what is being tested and the test's goals. The plan should explain what data will be collected, what should be assessed by testers (functions, tasks, etc.) and what you would like included in any feedback or bug reports (OS information, screenshots, etc.). The plan should also outline how beta testers should report the results of the test. If you don't openly let users know that you are seeking feedback, you run the risk of not getting any feedback.

6. Look Beyond Beta

During the time of mainframe coding, beta testing was a sufficient testing solution for companies. Using beta testers, a company could likely verify that there were no glaring errors remaining. Beta testers didn't have to extend themselves too much, as there were a limited number of bugs and problems that could occur. The testing process in general was more simplistic.

This is no longer the case. Now technology is accessible on countless devices across the globe. The number of problems and bugs that could arise multiplies with every new device, and every new international user.

In addition, the expectation of the user has changed. In the past it was expected that software would run slowly taking a few minutes to load and might not always work. Now, users expect an application to upload instantly and run at a real-time speed. Crashes are unacceptable and customers will give up entirely on a software or application that is slow or buggy. End-users expect a perfect application.

Because of this, it is no longer enough to leave testing up to beta testers. There aren't enough resources or control to ensure quality. While a beta might make improvements, it cannot fully ensure that the software will work flawlessly when it reaches customers.

Where is Beta Testing Headed?

Beta testing has evolved substantially, but it is a lot of work and doesn't guarantee quality. By using a beta test, you aren't testing with professionals and are leaving your test up to chance. While beta testing will likely never go away completely, other methods of testing offer guaranteed quality. Crowdsourcing in particular is an emerging, more reliable, direct

> **"The key to successfully employing a crowdsourcing model in a professional service category like web and mobile app testing is to advance beyond the loosely affiliated, unstructured crowd – a mob."**
>
> - Doron Reuveni
> CEO, uTest

competitor to beta testing. It can in some ways be compared to focused beta testing but avoids the common pitfalls of beta testing.

Crowdsourcing enables firms to tap into the creativity and diversity of a global community of skilled professionals. With crowdsourced testing, companies can select testers who match their target audience, while knowing that the people doing the testing are professionals. Like beta testing, crowdsourcing also gives companies the ability to test in real world situations. Through crowdsourcing, QA professionals take apps into real world, "in-the-" situations where they have the best opportunity to encounter potential problems.

The most common pitfall encountered when beta testing is the issue of getting quality feedback. With a good crowdsourcing company, testers are prepared and thoroughly trained professionals who have experience hunting for bugs, hitches and other vulnerabilities. They also know how to properly prepare a bug report and are accustomed to including screenshots and reproduction steps with each report – making the development team's job much easier than when they have to decipher a beta testing report that simply says "it crashed."

Beta testing offers a variety of benefits that in-house and outsourced testing simply cannot accommodate – namely testing with real users in real-life situations. But as the trend of users expecting applications to work perfectly from the moment the app reaches their hands continues to gain momentum beta testing will no longer be enough.

Companies can gain insight into how a market will respond to their product and discover fringe bugs with a beta test, but they'll have to decide if potentially angering highly demanding end-users (who often don't understand the true point behind a beta release) is worth the risk.

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 80,000+ 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** or **blog.utest.com**, or you can watch a brief online demo at **www.utest.com/demo**.

uTest, Inc.
153 Cordaville Road
Southborough, MA 01772
p: 1.800.445.3914
e: info@utest.com
w: www.utest.com