centercode

Beta Testing ROI

Applying Cost of Quality Principles to Beta Tests



A Whitepaper by Centercode

v1.2

1. INTRODUCTION

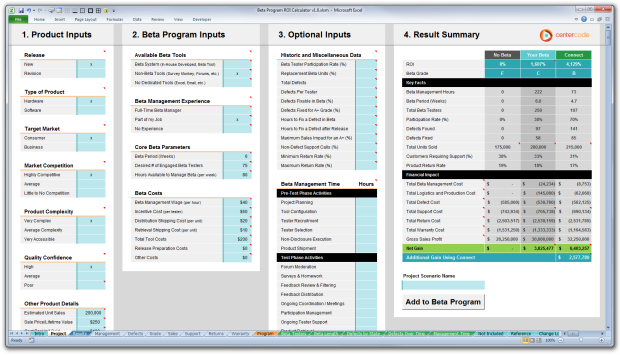
The goal of this whitepaper is to provide a better understanding of the economics of beta testing[1], particularly the Return on Investment (ROI). Without this understanding, beta tests are often seen as a "necessary evil," rather than a first-class contributor to a product's overall profitability. Yet, there are almost no resources available on the subject. This document offers a much-needed guide to building the business case for beta testing expenditures.

Calculating the ROI of beta tests is a challenging exercise. Many product and beta management teams have difficulty completing it because of two issues: 1) the lack of a common methodology for thinking about beta's costs and benefits and 2) the complexity of obtaining the right quantitative data.

This document solves both issues, not perfectly, but well enough to enable you to arrive at a defensible ROI figure. As with any ROI exercise, we urge you not to expect absolute accuracy. But with conservative estimates and multiple models, including best-case and worst-case scenarios, you'll find that beta testing often has such a large ROI that estimation errors pose minimal risk.

BETA TEST ROI CALCULATOR

To assist you in utilizing the model outlined throughout this whitepaper, we've developed and bundled a set of comprehensive Excel spreadsheets (also available at www.centercode.com/roi). The first is intended to identify the ROI of a single beta test (**Beta Project ROI Calculator v1.0.xlsx**), while the second is a macro-enabled[2] version (**Beta Program ROI Calculator v1.0.xlsm**) designed to identify the value of a complete annual beta program.



These spreadsheets are designed to help you achieve two goals: 1) assess the total ROI of your own beta test or program, as you've designed it, and 2) assess the additional value you can capture by using the **Centercode beta test management platform**.

Our calculators work based on a variety of inputs from you in conjunction with assumptions that we've provided based on our experience, research, and feedback from professional beta and product managers at leading technology companies. Or, you can override our assumptions with your own historical data to calculate a highly customized ROI. We've also created a video walk-through for our spreadsheet, available at www.centercode.com/roi/tour.

1 We use the term "beta testing" broadly to include all types of customer-based testing regardless of their names. So this document applies to user acceptance testing, field trials, customer validation, prerelease programs, and more. We also try to encompass all types of products, including hardware, software, enterprise, consumer, gaming, mobile, and web.

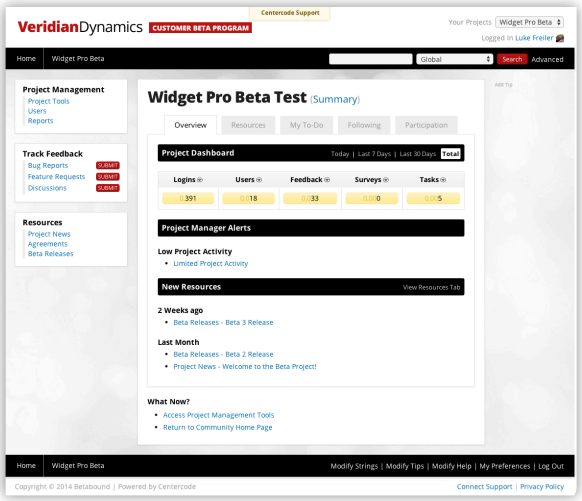
2 The macro-enabled version utilizes VBScript to copy individual beta test scenario models into a program worksheet, allowing for a more complete Beta Program ROI model. Functionally, this is a superior worksheet, but requires that you enable macros.

ABOUT CENTERCODE

You might find yourself wondering who we are and why we're writing this. Centercode is the industry leader in beta test management solutions. With more than a decade of experience delivering beta management software and services to some of the biggest brands in technology, we're a natural fit to produce these types of resources. In fact, this is just one of many free beta resources we've published to help companies improve their beta programs.

ABOUT OUR PLATFORM

The Centercode platform, our SaaS-based beta managment software, is built to handle every aspect of managing a beta program. Our platform helps you grow a community of highly profiled candidates, select the best possible testers, collect their feedback, share results, and much more. Best of all, it centralizes your entire beta program in one place.



Key ROI Benefits

- Shorter Beta Cycles
- Fewer, More Targeted Participants
- Far Less Time Invested
- Much More, Actionable Feedback

FEEDBACK AND QUESTIONS

If you have any questions, suggestions, or critiques about this whitepaper or the associated spreadsheets, please don't hesitate to contact us. Similarly, if you'd like help walking through the ROI of your own beta program, we'd be happy to assist you. While our business may revolve around our beta management software and service offering, our broader goal is to increase the success of beta programs everywhere.

Feel free to reach out in whichever way is most convenient to you.

Telephone: (800) 705-6540

Email: roi@centercode.com

Online: www.centercode.com

Twitter: @centercode

2. FORMAL THINKING ABOUT BETA ROI

A methodology for understanding the costs and benefits of beta testing is very useful for calculating the ROI. Reliable figures can be developed without a formal methodology, but a good methodology offers a framework and common vocabulary for classifying the inputs that determine your return. In other words, it makes beta ROI easier to conceptualize, discuss, and ultimately improve.

Because beta is often positioned between various functional groups, we've focused on finding an established methodology. This maximizes the possibility that useful cost data is already measured and available from other groups within the company. It also provides a body of academic and industry research to consult for guidance[3].

Specifically, we believe the Cost of Quality principles first developed in the 1940's and 50's — most famously by Joseph Juran (1951) — are a good choice for analyzing beta test ROI. The following sections introduce these principles and how they apply to beta testing.

A Note on the Intangible Benefits of Beta Testing

Defects often get the lion's share of attention in beta because they're very costly, and those costs make easy targets for correction. Other less quantifiable benefits of beta testing that you shouldn't overlook (even if you don't directly attach a dollar value to them) include: testimonials and user-generated content, press and analyst access, pre-launch customer face-time, documentation review, infrastructure testing, product review simulation, customer good will, and feature requests (market requirements).

Gathering feature requests is a very common function of beta testing. While our spreadsheet doesn't directly calculate return on feature requests, you can still choose to quantify them in your organization. This can be done by comparing the costs of managing feature request feedback against traditional market research efforts. Feature requests are a mostly free byproduct of beta testing, whereas traditional market research can be very expensive. Thus, to the extent that you can reduce market research expenditures based on beta feedback, beta has lowered your prevention costs in Cost of Quality terms and increased your ROI.

3 For an excellent survey of the history of Cost of Quality research, see Schiffauerova, A. and Thomson, V., "A Review of Research on Cost of Quality Models and Best Practices," International Journal of Quality and Reliability Management, Vol. 23, No. 4 (2006).

THE COST OF QUALITY FRAMEWORK

Cost of Quality (CoQ) principles grew out of a 1950's movement to improve the quality, efficiency, and profits of U.S. industry. Explosive growth in science, technology, and consumer demand required a new way of thinking about quality to keep pace with market changes while still producing satisfactory goods.[4] The CoQ principles advocated by Juran propose at least two critical definitions of quality. Quality can simultaneously refer to 1) a product's features and characteristics that influence customer satisfaction and 2) freedom from defects.[5]

In a general sense, the cost of quality is thought of as the cost of conforming to those definitions of quality, plus the cost of non-conformance.[6] In other words, it includes the costs invested in attaining a given level of quality, as well as the consequences of failing to meet expectations. Juran's more specific Cost of Quality takes this idea further, making CoQ the sum of the following four elements[7]:

> "COST OF QUALITY INCLUDES THE COSTS INVESTED IN ATTAINING A GIVEN LEVEL OF QUALITY, AS WELL AS THE CONSEQUENCES OF FAILING TO MEET EXPECTATIONS."

- **Prevention costs** are associated with preventing quality issues before they arise. They typically include investments in product planning and market research, audits of employees and processes, and training.
- **Appraisal costs** are incurred to evaluate the product's conformance with quality expectations prior to shipment. QA testing, product-line inspections, production audits, and beta testing are all examples of these costs.
- **Internal failure costs** are the costs of defects or non-conformance with quality expectations discovered before the product ships. They include, for example, redesign of hardware or software based on bugs found prior to release and the cost of scrap materials.
- **External failure costs** arise from defects or non-conformance after a product ships. Prime examples include support costs, product returns, warranty charges, and lost sales revenue.

COQ AND BETA TESTING

Now that the four elements of CoQ are understood, we can begin applying them to beta ROI. Beta testing is an **appraisal cost**, according to CoQ principles. It's a process that tests and validates the product's readiness for market by assessing its quality. However, companies often put too much emphasis on the second definition of quality (freedom from defects). Beta testing does serve that purpose, but it is also about evaluating how well the product satisfies customers' expectations (the first definition of quality).

4 Juran, J. and Godfrey, A., Juran's Quality Handbook, 5th Ed., Section 2.15 (1998).

5 Juran, J. and Godfrey, A., Juran's Quality Handbook, 5th Ed., Section 2.1-2.2.

6 Schiffauerova, A. and Thomson, V., "A Review of Research on Cost of Quality Models and Best Practices," p. 4.

7 Juran, J. and Godfrey, A., Juran's Quality Handbook, 5th Ed., Section 8.4-8.8.

Any purely mathematical calculation of beta ROI should begin by categorizing the appraisal costs associated with running a beta test. These include the labor time of those managing the beta, beta release engineering, logistical costs of distributing and collecting beta product, as well as any tool costs.

In the context of beta, **internal failure costs (IFC)** are the costs that result from beta test findings. For example, they include the cost of rework and regression testing for bugs discovered during beta. Because these costs are a result of beta, we consider them a cost of beta, even though remedying the issues after release may be unavoidable and more expensive. The gain from correcting issues before release, when they're less expensive and less damaging, is captured through external failure costs.

External failure costs (EFC) help us highlight the benefits of beta by pointing out the costs that were avoided because of beta testing. For example, by correcting defects during beta, you can avoid more expensive corrections after release, along with support costs related to those defects, and much more. This is a key reason beta testing should be thought of as more than an expense. An investment in beta testing can have a hugely positive impact on the product's bottom line. Product quality improvements resulting from beta can also enhance sales through word of mouth and product reviews, while reducing product returns and decreasing warranty reserve charges.

The process of quantifying these effects is addressed in Part 3, following our working definition of beta ROI.

A WORKING DEFINITION OF BETA ROI

We are defining the ROI of beta to mean the net gains from reduced EFC[8], divided by the appraisal costs of running the beta plus the IFC[9] of correcting discovered defects, multiplied by 100.

$$\frac{\text{Net Gains from EFC Reductions}}{\text{(Appraisal Costs of Beta Management + Rework and Retesting IFC)}} \times 100$$

Figure 2-1 illustrates the relationships between these costs. Let's say that a hypothetical company has an established QA program but runs no beta tests for its product. This company's initial CoQ will include some prevention costs, such as product planning. There are also significant appraisal costs and IFC, even without a beta, from investments in QA testing. Finally, elevated EFC are likely, because QA can rarely simulate real customer usage and environments.

Following implementation of beta testing, the company's CoQ will be noticeably different. Prevention costs will remain steady. Appraisal costs and IFC will both increase. But EFC will decline sharply. Overall, running beta tests has created cost increases, but total CoQ decreases are higher, by virtue of EFC savings. The final result is a positive ROI.

8 The net gains are the gains from reduced EFC, minus the cost of your investment (i.e., the costs from the denominator in the equation above).

9 For reference, our spreadsheet considers pre-release defect correction and support material creation as beta IFC



Figure 2–1

Figure 2-2 shows the relative value of two beta testing solutions, by comparing the CoQ expenditures for two approaches to beta testing. In this scenario, the new solution has similar tool costs but offers time savings for the beta staff, so it decreases appraisal costs. The second solution also helps increase participation from beta testers, which means discovering more bugs, so internal failure costs increase. However, this solution is likely to provide a proportionally larger decrease in external failure costs.

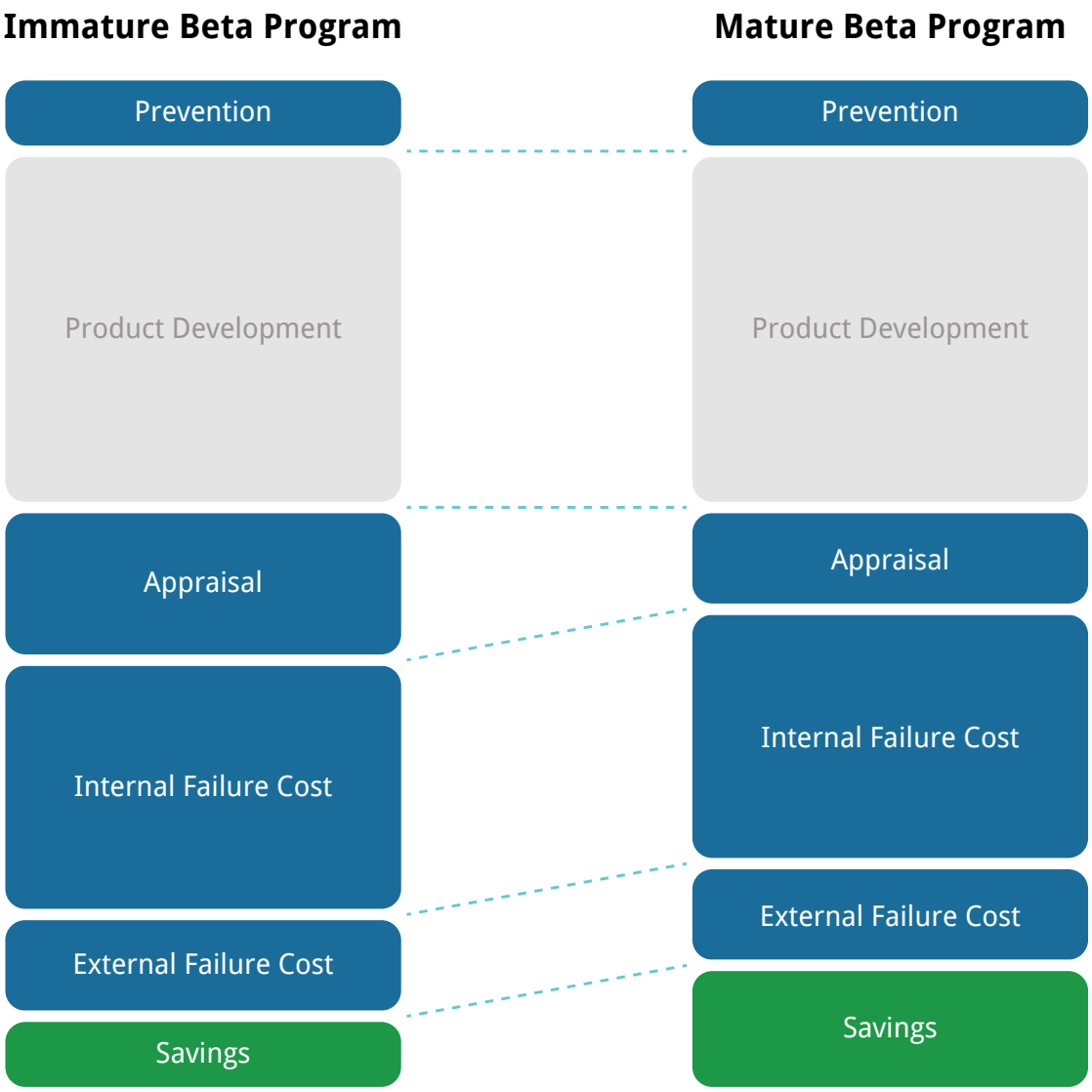


Figure 2–2

We like these diagrams for demonstrating the intuition behind beta ROI. To draw meaningful comparisons between beta solutions, however, we prefer to calculate ROI of each against the failure costs of running no beta. That's the approach we use in our spreadsheet, because we feel it's the only way to calculate a true comparison.

3. QUANTIFYING THE INPUTS AND RESULTS OF BETA TESTING

The biggest stumbling block that teams face when calculating beta ROI is obtaining data. Historical data does make the process easier and more accurate. However, there are other approaches. Data from secondary sources and conservative modeling can be used to create a justifiable ROI figure that illustrates the tremendous value of beta testing.

The remainder of this paper focuses on identifying the important data points in a beta ROI calculation, as well as data sources and methods of estimation. We'll point out where processes are different for hardware and software companies and introduce various models to help estimate necessary values, when hard data are not available. Basic arithmetic is all you need to use these models.

APPRAISAL COSTS

Almost the entire process of beta testing is an appraisal cost.[10] However, you must first identify each component cost of running a beta test before you can calculate the total appraisal cost. Specifics may vary from one company to the next, but the five main cost sources for running a beta test are: labor, production, logistics, incentives, and tools.

A. Labor

You can't run an effective beta test without investing employee time in its management. Whether that means one person or a whole team will depend on the scope of your beta testing. In either case, you should appropriately account for employee time and anticipate the variety and time requirements of beta tasks.

To account for an employee's real cost to the business, calculations should use the employee's fully burdened rate. That way, hidden and easily forgotten costs, like paid time off, employee benefits, and company-provided resources, are included in the calculation. For simplicity, we'll use a burden rate of 30% in the following calculations.

For example, let's start with one product manager making $75,000 per year who is tasked with running a beta test. The estimated total annual cost of that employee would be:

$$\$75{,}000/\text{year} \times 1.3 = \$97{,}500/\text{year}$$

To convert that total cost into an hourly rate, we divide the fully burdened cost by 2,080 hours (40 hours per week x 52 weeks per year). The fully burdened hourly rate would then be:

$$\$97{,}500 \div 2080 \text{ hours} = \$46.88/\text{hour}$$

To apply this to beta test ROI calculations, we need to know how much time the product manager devotes to beta testing. If we assume 40 hours over four weeks, then labor contributes $1,875.20 ($46.88 x 40) to appraisal costs.

10 An argument can be made that certain parts of beta testing (e.g., research conducted for use in planning the next generation of a product) should be classified as a prevention cost. To the extent that the ROI exercise is for an entire beta program, rather than a single beta test, this becomes increasingly true.

In practice, two or more people often contribute to beta test management across a wide variety of tasks. By estimating the ratio of time each person spends managing the beta, you can calculate a blended rate that provides a fair amount of precision. But for practical purposes, a simple average of the fully burdened rates will usually suffice.

Estimating the total amount of time employees will spend managing a beta test can be more difficult. To help you make these calculations, Table 3–1 identifies common beta test management tasks, along with the time required (**in minutes**) for each task, depending on your approach to beta (see **The Tools of Beta Testing** for additional context). Most beta tasks scale according to different factors or combinations of factors, such as the number of weeks spent testing, defects, total testers, and engaged (that is, participating) testers.[11]

Activity	Scales With	No Tools	Non-Beta Tools	Beta System	Centercode
PRE-TEST PHASE					
Project Planning	Project Scope	240	240	240	180
Tool Configuration	Tools	0	120	60	30
Tester Recruitment	Testers	5	5	4	1
Tester Selection	Testers	5	5	4	3
Non-Disclosure Execution	Testers	5	5	4	1
Product Shipment	Testers	6	6	5	4
BETA TEST PHASE					
Forum Moderation	Engaged Testers, Weeks	0	3	3	3
Surveys and Homework	Engaged Testers, Weeks	0	2	1	1
Feedback Review & Filtering	Defects	6	6	4	3
Feedback Distribution	Defects	5	5	2.5	0
Coordination / Meetings	Weeks	240	240	120	120
Participation Management	Testers, Weeks	4	4	3	2
Ongoing Tester Support	Engaged Testers, Weeks	5	4	3	2
Product Retrieval	Testers	5	5	5	5
Incentive Management	Testers	5	5	5	1

Table 3–1

11 This table is also available as a worksheet in our Beta ROI spreadsheet, which will calculate the scaling values based on your inputs.

B. Production

For hardware companies, manufacturing beta products is one of the most costly areas of beta testing, because small-batch, pre-release manufacturing is disproportionately expensive. Fortunately, these costs can be easily obtained for each unit, either in-house from manufacturing or from a purchase order, if outsourced. Then, the beta team must evaluate how many units are required.[12]

In some cases, hardware requirements are as simple as one unit per beta tester, but other factors can come into play. Often times, additional units are requested as a safeguard against hardware failure during testing. Increasing the number of units by 10% to 20% is typical. If financial constraints limit the number of available units, testing can be split into multiple phases with different teams of testers. After the first phase, the beta units are retrieved and redistributed for the second phase. However, this approach significantly increases the length of the beta test.

"OFTEN TIMES, ADDITIONAL UNITS ARE REQUESTED AS A SAFEGUARD AGAINST HARDWARE FAILURE DURING TESTING."

On the software side, many companies believe there are no beta unit production costs, other than fringe cases involving physical media. But what if a software release engineer spends time working on one or more beta builds during testing? That time should be factored in as a production cost, at the engineer's fully burdened rate. Essentially, any time spent planning, messaging, building, and supporting beta specific software releases and patches should be allocated to beta testing in some form.[13] This principle holds true for the software powering hardware products as well.

C. Logistics

Logistics, like production, is a cost category that generally impacts hardware companies more than software companies. Beta testing a physical product always involves a baseline logistical cost for delivering that product to testers (overnight delivery being the best practice). Often, the product must be returned after testing, as well. And to ensure the highest possible compliance, the company must pay for return shipping. In addition, any costs for shipping and retrieving replacement units must be accounted for.

"IF YOU'RE USING A CONTENT DELIVERY NETWORK FOR BETA SOFTWARE, THIS SHOULD BE CONSIDERED A LOGISTICAL COST."

Software is generally distributed over the Internet now, so logistical costs in software betas are often minimal. In rare cases, beta versions of very large software projects will be distributed over Content Delivery Networks (CDNs) like Akamai. In that case, CDN fees should be considered a logistical cost.

12 It's common for manufacturing to determine available units for beta testing, rather than the beta team deciding how many are needed.

13 For ease of calculation, these costs can be either Production Costs or IFC. (Technically, they should be Production Costs for the initial beta release and planned subsequent releases but IFC for releases necessitated by discovered defects.) For ease of calculation, however, either is fine.

D. Incentives

The most common incentive for beta testers is to reward active participants with the tested product at the end of the beta[14]. To account for the cost of these incentives, some companies use the production cost of the product as their metric. It concretely represents what the company spent to furnish that incentive, especially for hardware products. Other companies believe that the opportunity cost of not selling the retail product to the tester is a better choice. This approach captures potential lost revenue, but it also assumes that everyone testing would have purchased the product.

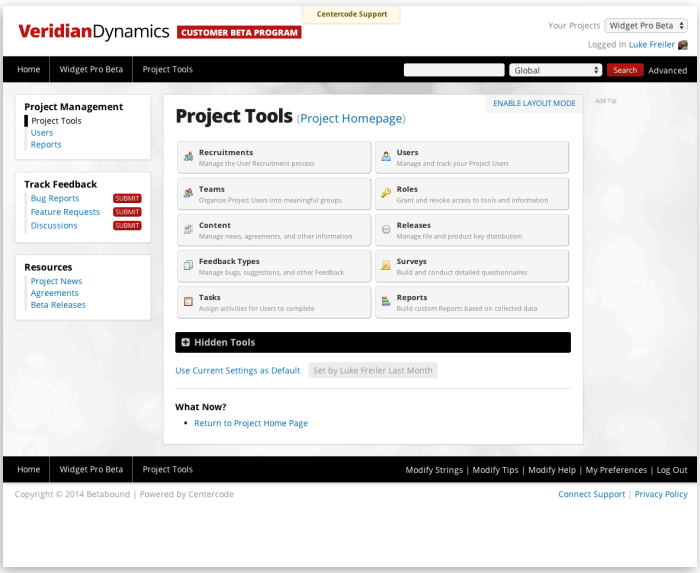
> "THE MOST COMMON INCENTIVE FOR BETA TESTERS IS TO REWARD ACTIVE PARTICIPANTS WITH THE TESTED PRODUCT AT THE END OF THE BETA."

The best option is to decide incentive accounting on a case-by-case basis. However, for most hardware products, your production costs are likely to be higher than your hypothetical lost revenue. For software products with "zero" production costs (or close to it), it's far less accurate to say there is no cost to providing the software for free, especially with a large tester pool.

Other incentives, such as gift cards for major retailers like Amazon, are also popular, and the costs are clear.

E. Tools

Tool costs in beta are more straightforward. If tools are used solely for beta testing, then the full cost should be applied in the ROI calculation. When beta testing is just one of many contexts in which the tools are used in, the cost should be based on the percentage of the tool's total use attributable to beta testing. Finally, don't add costs for software tools, such as Microsoft Office, where seats are licensed for everyone in the company. Those costs are typically accounted for in an employee's fully burdened rate.



To find out more about how beta test management tools can impact (and improve) your own beta ROI, see the **Improving Your Beta ROI** section.

14 Hardware companies often want the beta product returned, to be replaced by a final version, because pre-production units don't necessarily present the company in the best light.

The Tools of Beta Testing

In the process of developing our ROI spreadsheet, we considered the various ways that companies manage their beta tests and the relative impact on ROI. We've found that, more than anything, choice of tools defines the maturity and capabilities of a company's beta testing. Those tools can be broken down into the following four categories:

1. **Productivity Tools** Many companies manage their betas with standard desktop business software and services like MS Word, Excel, PowerPoint, and email. This is often the result of an ad hoc decision to beta test or inexperience with the demands of beta.

   Pros: No new cost, already available

   Cons: Poor customer experience greatly reduces participation, difficult to organize, difficult to distribute without information falling through the cracks, impossible to scale

2. **Non-Beta Software & Services** The most common approach to beta is using an assortment of single-function tools that, collectively, approximate a complete beta testing experience. These tools typically include user forums, online surveys, customer-support platforms, and defect-tracking systems. This approach exhibits a fuller understanding of beta requirements, but lack of resources or a low priority on beta prevent a beta-specific system from being used.

   Pros: Inexpensive, easily accessible, many choices

   Cons: Utilizing multiple systems is disjointed for testers (reducing participation), difficult to manage, produces fragmented data, participation difficult to track

3. **In-House Beta System** Larger companies that have significant beta programs often rely on internally developed systems designed specifically for beta testing. These systems provide a unified and secure interface for the common elements of beta, including feedback (bugs and requests), user forums, and surveys. Because of the resources required, these systems are found almost exclusively in organizations with mature beta processes and dedicated teams.

   Pros: Built for beta, customized for your processes/needs, dedicated experience for testers reduces friction and increases feedback, may include other system integrations

   Cons: Immensely expensive to develop and maintain, the platform doesn't evolve without ongoing investment, risks going end-of-life if internal support wanes

4. **Commercial Beta Test Management System (e.g. Centercode)** These systems are developed and offered specifically for the purpose of managing beta tests. They excel at the basic requirements of beta, constantly incorporate innovative features that save time and effort, and implement more advanced functionality as beta testing evolves. Companies focused on maximizing beta results via established beta programs gravitate toward these solutions.

   Pros: Built for beta, economies of scale, highly positive tester experience, better feedback organization, flexible to meet evolving requirements, integration with existing tools

   Cons: Niche offering with few vendors, likely more costly than single-function tools

INTERNAL FAILURE COSTS

Internal failure costs (IFC) stem from quality issues discovered prior to a product's release. In beta testing, this typically means rework and regression testing required for bugs discovered by beta testers. It might also include creation of support materials for known defects at release. By themselves, these costs don't represent a return on the company's investment in beta testing. In fact, they increase costs associated with the beta project by shifting product fixes to an earlier point in the process. However, the improvements they yield can bring about very significant reductions in external failure costs. In that sense, substantial beta ROI requires some degree of IFC.

IFC from beta can be calculated in a variety of ways. We've selected a relatively simple model based on the following four parameters, which can be easily obtained or estimated: 1) the number of defects corrected in beta testing[15], 2) the average number of hours required to fix and test defects found during beta, 3) the fully burdened rate of all engineers performing the rework, and 4) the fully burdened rate of all QA team members testing the fix.

The number of defects, n, and the average number of hours, h, are multiplied to estimate the total time devoted to correcting internal quality failures. Then we use the simple average of the two wage parameters to address the differing costs of rework and testing. The total hours multiplied by this average wage yields the IFC of rework and regression testing, as shown by the following formula:

$$n_{defects} * h_{avg} * [(\text{fully burdened rate}_{engineering} + \text{fully burdened rate}_{QA}) / 2]$$

If you don't have defect data to work with, here are some insights from tests run by our own Managed Beta services group. In a sample of 30 recent beta tests, comprised of both hardware and software products, the median number of defects observed was 41. There is a noticeable correlation between the number of testers and the number of defects reported. However, less than half the variation in the number of defects can be attributed to the number of testers ($r^2 = 0.475$). The quality of the product and length of testing also play important roles in defects reported.

When estimating the average hours required to fix a beta defect, it's useful to remember Barry Boehm's maxim that failure costs increase steadily through each phase of a project. For example, if a defect is found in the unit testing phase, it will generally require much less work to correct than if the same defect is found, fully integrated into the product, during beta. A 2002 research study from the National Institute of Standards & Technology (NIST) provides some context for this principle. It reported that the average hours to fix defects uncovered during beta testing ranged from 6 to 15 in the industries surveyed. For defects discovered during unit testing, the range was only 2 to 8 hours.[16]

Other sources of common IFC include documenting release notes, changes to specs and marketing messages, and—occasionally—delaying a product release due to critical defects.

15 For a variety of reasons, such as time, complexity, and frequency, some bugs discovered in beta testing will not be fixed by product release.

16 NIST, "The Economics Impacts of Inadequate Infrastructure for Software Testing" (May 2002). Available at http://nist.gov/director/planning/upload/report02-3.pdf.

EXTERNAL FAILURE COSTS

Reduced external failure costs (EFC) are where companies generate an actual return on their investment in beta testing. By spending money to evaluate quality and fix defects internally, companies can avoid a wide range of expensive external quality issues. Most significantly, they experience: 1) less post-release defect correction, 2) decreased revenue lost to product returns, 3) lower support costs, 4) smaller warranty reserves, and 5) fewer lost sales due to negative product reviews and word of mouth. These five areas are key to preserving product revenue and not wasting market opportunities.

"REDUCED EXTERNAL FAILURE COSTS ARE WHERE COMPANIES GENERATE AN ACTUAL RETURN ON THEIR INVESTMENT IN BETA TESTING."

A. Post-Release Defect Correction

This topic extends the earlier IFC discussion, since the defects you discover and correct during beta lower or eliminate costlier efforts to correct those defects after release. That savings is an integral part of calculating beta ROI. But because our ROI formula categorizes early defect correction as an IFC, it affects what we calculate here. Instead of determining the difference between pre- and post-release costs, we will calculate the entire post-release cost avoided.

We will describe two ways to present this information. The more straightforward method is to reuse the formula from our discussion of IFC.[17] The only parameter that changes is the number of hours to correct the defect, reflecting the increased difficulty of repairing a fully integrated, released product. The number of defects remains the same. And unless you're basing the wage calculation on something more complex than an average of engineering and QA salaries, that figure will also stay the same. Your result then would look something like this:

40 bugs * 20 hours * $60/hour = $48,000 avoided in post-release corrections[18]

Using this method, it's easy to compare the costs of defect correction during beta and after release. However, the second method of presentation—which we've adopted for our spreadsheet—adds additional context that helps highlight the value of beta.

With this second method, you start by estimating a total number of defects in your product as it enters beta. Then you can calculate the total cost of correcting those defects post-release and compare it with the reduced cost after beta has enabled earlier corrections, as shown in the following example:

450 total bugs * 20 hours * $60/hour = $540,000

410 bugs (40 corrected in beta) * 20 hours * $60/hour = $492,000

$540,000 - $492,000 = $48,000 avoided in post-release corrections

17 $n_{defects} * h_{avg} * [(\text{fully burdened rate}_{engineering} + \text{fully burdened rate}_{QA}) / 2]$

18 The corresponding calculation for beta might be: 40 bugs * 20 hours * $60/hr = $36,000.

As you can see, the calculations work out the same. But this second presentation emphasizes the magnitude of costs that might still be decreased if more resources (tools, time, testers, and so forth) were invested in beta testing.

If you need data on the time to fix defects after release, sources vary greatly. The 2002 NIST report mentioned earlier reports a range of 10 to 18 hours in the industries surveyed. In 2008, IDC surveyed 139 companies in North America about software quality and arrived at 30 hours as a benchmark for correcting post-release defects. Specifically, nearly 50% of the respondents said the fixes take between 2 and 5 workdays, on average.[19] Approximately 30% reported that they took more than 5 workdays. Unless there is important evidence to the contrary, 20 hours seems to be a reasonable option for reconciling these differences. However, if you have a particularly complex product, that figure could increase significantly.

B. Product Returns

In the United States, the total value of products returned every year is estimated at $100 billion[20], making returns a significant source of lost revenue. Beta testing can have a noticeable impact on product returns, but quantifying that effect is a considerable challenge. This section explores how product returns cost companies money, and it proposes an approach for estimating beta testing's effect on the returns of specific products. This figure can then be used in ROI calculations representing the decrease in EFC stemming from product returns.

This information applies mainly to hardware companies. Return policies and options in the software world are much narrower, where they exist at all. And, as a result, the types of scenarios that lead to hardware returns instead generate additional support incidents for software. However, our product return model could be simplified for software returns where appropriate.

"IN THE UNITED STATES, THE TOTAL VALUE OF PRODUCTS RETURNED EVERY YEAR IS ESTIMATED AT $100 BILLION."

The degree to which a product return results in lost revenue depends largely on what can be done to recapture its value afterward. You can think of the product return stream as a kind of leaky funnel, according to Blackburn et al.[21] The first (and best) outcome for a returned product is restocking. Restocking the product as new allows the company to resell it at or near its original retail price. The next best outcome is low-touch refurbishing, followed by high-touch remanufacture, component salvage, and scrap. Figure 3–1 illustrates the return stream, as well as the relative likelihood of each outcome, according to their research.

19 Ballou, Melinda-Carol, "Improving Software Quality to Drive Business Agility," IDC (June 2008). Available at http://www.coverity.com/library/pdf/IDC_Improving_Software_Quality_June_2008.pdf.

20 Stock, Speh, and Shear, "Many Happy (Product) Returns," Harvard Business Review (July 2002). Available at http://hbr.org/2002/07/many-happy-product-returns/ar/1.

21 Blackburn, Guide, Souza, and Wassenhove, "Reverse Supply Chains for Commercial Returns," California Management Review, Vol. 46, No. 2 (Winter 2004).

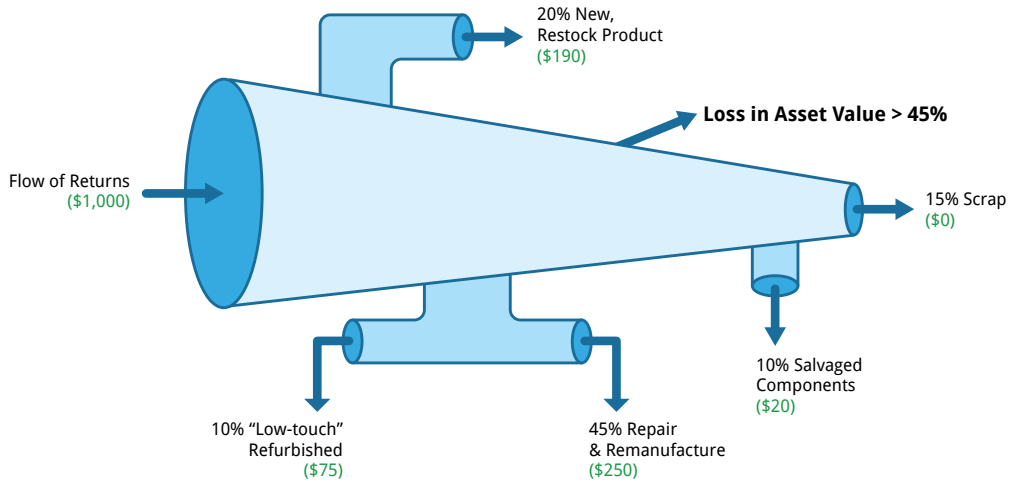


Figure 3–1

While these values are helpful, they may not be accurate for all products or industries. A company that meticulously tracks product returns might have historical data that offers better insight into the likelihood of each outcome and its corresponding loss in product value. When starting from scratch, however, it's hard to see how these ideas can be reduced to a dollar figure representing the amount of money beta testing can save on product returns.

We believe that breaking down the problem into components that can be estimated more easily and intuitively allows these figures to be estimated. The first stage is to address the likelihood of each outcome in the return stream. Unfortunately, estimating how many units will be restocked rather than refurbished or remanufactured is difficult without additional context. But if we look one step beyond—to the reasons that customers return products—a clearer picture emerges. If the customer returns the product because of buyer's remorse, then the product will probably be restocked or refurbished on a low-touch basis. Because the product wasn't returned for quality reasons, there's no need to remanufacture, salvage, or scrap it. On the other hand, if a customer claims that a unit is defective out of the box, restocking is unlikely.

Accenture recently published a detailed study on product return inefficiencies[22] in the consumer electronics industry, and this report breaks product returns into three categories: Buyer's Remorse, No Trouble Found, and Defective Products. Buyer's Remorse generally consists of various convenience-related returns, such as actual remorse, finding a better price elsewhere, or essentially renting the product.[23] No Trouble Found returns result when a product fails to meet the user's expectations, is too complex or confusing, or there are unverified hardware

22 Douthit, Flach, and Agarwal, "A Returning Problem: Reducing the Quantity and Cost of Product Returns in Consumer Electronics." (2011) Available at http://www.accenture.com/SiteCollectionDocuments/PDF/Accenture-Reducing-the-Quantity-and-Cost-of-CustomerReturns.pdf

23 Souza, Guide, Wassenhove, and Blackburn, "Time Value of Commercial Product Returns," Industry Studies Association Working Papers (2004).

or software issues (real or perceived by the user).[24] The report found that buyer's remorse makes up 27% of product returns and No Trouble Found scenarios represent another 68%[25], which means only 5% of returns are verifiably defective.[26]

If we use these three categories for our ROI estimates, we can assign each outcome, such as restock or refurbish, a probability specific to that type of return. This allows us to create a conditional probability table (or conditional distribution), as shown in Table 3–2.

	Restock	Refurbish	Remanufacture	Salvage	Scrap
Defective Product	.05	.33	.33	.15	.14
No Trouble Found	.33	.4	.2	.05	.02
Convenience Return	.8	.2	0	0	0

Table 3–2

Next, we can assign reasonable probabilities for the return categories themselves and multiply them by the conditional probability values. This creates a joint probability distribution *j* that gives the probability of occurrence for each set of events. From there, we can calculate the marginal probability of each outcome by adding up the columns. The marginal probabilities can ultimately be substituted for the general data provided by Blackburn et al. Table 3–3 shows examples of joint and marginal probabilities.

	Restock	Refurbish	Remanufacture	Salvage	Scrap
Defective Product (p = 0.5)	.01	.066	.066	.03	.028
No Trouble Found (p = 0.68)	.09075	.11	.055	.01375	.0055
Convenience Return (p = 0.27)	.1	.025	0	0	0
Marginal Probability	**.30075**	**.401**	**.181**	**.05**	**.06725**

Table 3–3

24 Douthit, Flach, and Agarwal, "A Returning Problem," p. 3.

25 If this number seems high, it's because it is. No Trouble Found returns and non-defect support calls are very expensive for companies. In fact, one of the biggest trends in beta today is to use beta for testing the customer out-of-the-box experience. Failing to get that experience right can single-handedly sink a product's profitability.

26 Ibid. p. 6.

The next step is to determine the value recaptured through each of the five potential outcomes. In the leaky-funnel model discussed earlier, the value recaptured for each outcome is 95%, 75%, 55.5%, 20%, and 0%, respectively. A weighted average can then be calculated by multiplying the marginal probability of each outcome by its corresponding recaptured value and taking the sum of those products, as shown in Table 3–4.

	Marginal Probability	Value Recaptured	Weighted Value
Restock	.30075	.95	.28571
Refurbish	.401	.75	.30075
Remanufacture	.181	.55	.09955
Salvage	.05	.2	.01
Scrap	.06275	0	0
		Weighted Average	**.69601**

Table 3–4

Finally, to calculate the value *lost* from product returns for a given joint distribution *j*, we first determine the product of estimated units sold (*s*), sale price (*p*), and return rate (*r*). Then we subtract from it the product of estimated units sold (*s*), sale price (*p*), return rate (*r*), and weighted average value recaptured (v_r). That formula looks like this:

$$spr - sprv_r$$

At this point, we know how much value is lost to returns for one particular scenario—ideally a scenario in which no beta test was conducted. At least one more calculation[27] must be performed to measure the impact of beta testing. This process works the same way, but different values are substituted for the probability of defective product returns, No Trouble Found returns, and convenience returns. The dollar value of the beta testing effect (i.e., the contribution of returns to EFC reduction) is the difference between the $spr - sprv_r$ calculations for each distribution.

Keep in mind that the same conditional probability table can be used for both joint distributions, and overall return rates can vary from high single digits to 35%, depending on the type of product[28]. Finally, you may notice that our spreadsheet scales return rates and reasons with the beta test grade. That's because our ROI calculations have to respond algorithmically to a wide range of user inputs instead of fixed data points.

27 In our spreadsheet, we perform this exercise three times. Once for no beta test, once for a beta test given your available tools, and once for a beta test run with the Centercode beta test management platform.

28 Sauza et al.

C. Support Costs

The effect of beta testing on support costs can be determined using the same techniques discussed under product returns. The key inputs are: 1) volume of product sold; 2) percentage of customers affected by defects; 3) the distribution of defect severity; for example, critical, minor, trivial, and non-defect[29]; 4) the level of support required to fix bugs in each severity category, such as self-support, Level 1, Level 2, and so forth; and 5) the average cost per incident incurred at each level of support.

When creating the model, a useful starting point is the third input: distribution of defect severity. Using historical data or estimates, start by creating a probability distribution for the different categories of bugs that your organization tracks, as shown in Table 3–5.

Description	Probability
Critical	.1
Minor	.3
Trivial	.4
Non-Defect	.2

Table 3–5

Next, for each category of defect, assign probabilities for the different levels of support resolution. For trivial and non-defect issues, the probabilities may be heavily skewed in favor of self-support and Level 1 support. But for critical bugs, you're more likely to see Level 2 and Level 3 support. Table 3–6 provides an example.

	Self-Support	Level 1 Supp.	Level 2 Supp.	Level 3 Supp.
Critical	.1	.2	.4	.3
Minor	.4	.5	.1	0
Trivial	.8	.2	0	0
Non-Defect	.4	.5	.1	0

Table 3–6

By multiplying each value in the Table 3–6 with the probability of its corresponding defect category, you create a joint probability distribution *j*. Then you can calculate the marginal probability of each support resolution level by taking the sum of each column as shown in Table 3–7.

29 Non-defect refers to issues like customer confusion that lead to support calls without being actual defects.

	Self-Support	Level 1	Level 2	Level 3
Critical (p = .1)	.01	.02	.04	.03
Minor (p = .3)	.12	.15	.03	0
Trivial (p = .4)	.32	.08	0	0
Non-Defect (p = .2)	.08	.10	.02	0
Marginal Probability	**.53**	**.35**	**.09**	**.03**

Table 3–7

Next, multiply the marginal probability of each support resolution level by its corresponding cost per incident. When you add up the resulting values, you create a weighted average cost per incident (c_i) as shown in Table 3–8.

Support Level	Marginal Probability	Cost per Incident
Self-Support	.53	N/A
Level 1 Support	.35	$20.00
Level 2 Support	.09	$50.00
Level 3 Support	.03	$100.00
	Weighted Average	**$11.52 per incident**

Table 3–8

The product of this weighted average, volume of product sold (v), and percentage of users affected by defects (u_d) gives you an estimate for variable support costs. By adding fixed support costs (c_f), like the creation of knowledge base materials, you then have a total support cost.

$$\text{Total Support Costs} = c_i * v * u_d + c_f$$

Your next step is to calculate the change in Support Costs brought about by beta testing. One method is to simply perform the previously outlined steps twice, adjusting the probabilities for the impact of beta testing, then determining the difference. For example, after beta testing, the probability of critical bugs reaching Level 2 and Level 3 support might decrease by 20%, because the bugs were discovered during beta, and the support team is better prepared to assist users. A similar shift toward self-support may also occur for lesser defects (20% to 30%). And, overall, you'd hope to see fewer users affected by any defect, because more defects were discovered and fixed as a result of improved testing. This increase in fixes can be up to 20% for highly effective betas.

In our spreadsheet, we take a slightly different approach. We estimate not only the number of defects found, but also the number corrected. Then we use the number of uncorrected defects to adjust support costs. After all, you can and should still take action to reduce the impact of defects that you found in beta but didn't fix. We assume that half the uncorrected defects can now be resolved through self-support, while support staff can be better prepared for the other half and resolve those incidents at Level 1. As a result, we reduce the probability of Level 2 and Level 3 support by the percentage of found/unfixed defects, and we redistribute the reductions to increase the probability of self-support and Level 1 support. This method illustrates that successful beta testing can result in higher fixed support costs, but lower variable support costs, which produces a decrease in total support costs.

D. Product Warranties

Product warranties vary greatly, so it's difficult to provide specific insights for measuring their effect on beta testing ROI. However, under Generally Accepted Accounting Principles, product warranty expenditures are considered contingent losses that are charged against income to the extent that they can be reasonably estimated. In other words, when a product is sold, the company must hold a portion of that sale income in an account to cover estimated warranty costs for the duration of the warranty. When warranty costs are incurred, the amounts are deducted from this account. According to Accenture research on warranty reserve management, these reserves average around 2.5% to 4.5% of revenues.[30]

This charge against income can tie up substantial sums for lengthy periods, if a product comes with broad warranty provisions. For software companies, this may not be important. Their warranties tend to be so narrow and so short that beta testing may have little or no impact on the contingent liability estimate. For hardware companies, it's worth investigating the terms of the product's warranty and discussing the impact of beta testing with the accounting department. Consider the case of a product with $50,000,000 in gross sales. If beta testing reduces estimated warranty liabilities by .5%, that's $250,000 attributable to beta testing ROI.[31]

"ACCORDING TO ACCENTURE RESEARCH ON WARRANTY RESERVE MANAGEMENT, THESE RESERVES AVERAGE AROUND 2.5% TO 4.5% OF REVENUES."

E. Sales (Product Reviews and Word of Mouth)

Perhaps the most difficult EFC change to quantify is the effect that beta testing has on sales that result from product reviews and word of mouth. Experience tells us that products with quality issues tend to have lower reviews on web sites like Amazon.com, which customers often view before they make a purchase. We also know that beta testing allows many hidden quality problems to be corrected before release. Unfortunately, it's very difficult to measure the effect on sales, even though we intuitively know that one exists.

30 R. Douglas Derrick, "Rethinking warranty management to achieve high performance and differentiation" (2004). Available at: http://www.accenture.com/SiteCollectionDocuments/PDF/rethinking_usltr.pdf

31 It might also be appropriate to factor in the time value of money and compound the estimated amount by a given interest rate for the number of years covered by the warranty. Again, the best option is to consult the company's accounting department.

According to Cone Communications, however, 69% of U.S. adults actively research products and services online before purchase. Eighty-five percent were more likely to buy a product when they could find additional recommendations online, and 64% specifically searched for consumer or user reviews before purchasing. Finally, 87% said that positive online information reinforced their purchase decision, and negative information caused 80% to change their mind about a product.[32]

These findings suggest that, if you sold 100,000 units of your product and it had glowing reviews, those reviews impacted about 60,000 of your sales. But if the product had terrible reviews, you might have missed out on an additional 81,000 units sold. There's clearly a large margin of error in these examples. Some people would have purchased the product anyway without the good reviews, while others likely purchased it despite bad reviews. But the findings still illustrate the effects of product quality on today's savvy shoppers.

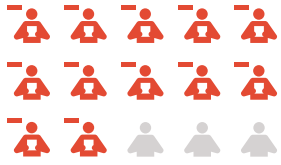
To our knowledge, there's no proven method for quantifying the effects of beta testing on product sales. However, data like the Cone Communications study can be used to support conservative estimates. You should also consider the competitiveness of your market and the nature of your customers. In a highly competitive market, poor quality may have a large detrimental effect, while good quality is rewarded with smaller gains. In a market with low competition, quality may not have a significant effect on either end, because substitute goods are scarce. As for customers, reviews about product quality appear to be more pervasive and influential for consumer customers than enterprise. We have built this reasoning into our ROI spreadsheet to calculate the beta effect as a percentage of sales influence determined by the grade of your beta test.



69% of U.S. adults actively research products and services online before purchase



87% say that positive online information reinforced their purchase decision



80% change their mind about a product because of negative information online

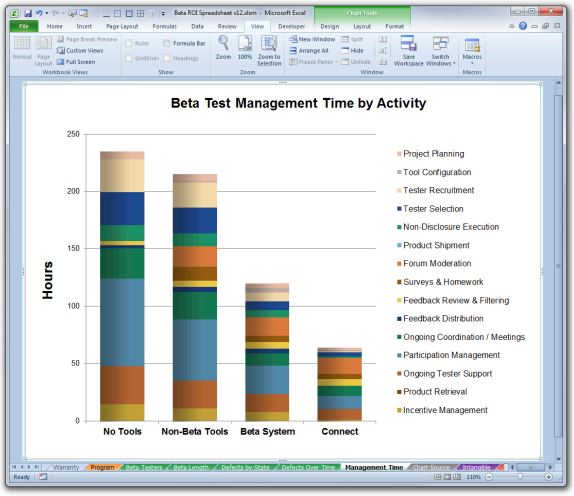
32 2011 Cone Online Influence Trend Tracker.

IMPROVING YOUR BETA ROI

Once you know how to calculate beta testing ROI, the natural inclination is to start improving that ROI figure. Luckily, Cost of Quality categories offer clear targets for improvement. You look for ways to adjust your appraisal costs and IFC, while preserving or maximizing the benefits of beta testing (your reduction in EFC). The following sections highlight some common methods for improving your beta ROI.

A. Beta Management Time

First, look at the amount of time devoted to managing your tests. The less well-suited your tools are for managing beta testing tasks, the more inefficient your beta process becomes. For example, if you use email and spreadsheets to manage beta tests, executing NDAs becomes a manual process. Or consider your defect reporting. If tester forums aren't built for beta, how much time will you spend cataloging and linking every bug post to similar reports and cross-referencing the users' test platforms? With Centercode, we greatly reduce beta management time by automating time-consuming tasks like NDA management and participation recruitment, while helping you build reusable resources like project templates and grow a persistent community that's ready to test on short notice. As a result, Centercode beta tests require a fraction of the time investment, and at $40 or more per hour for beta managers, those savings add up quickly.



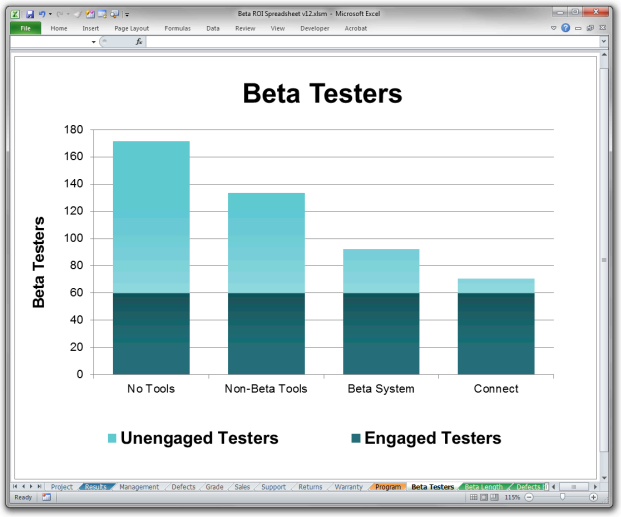
B. Feedback Workflow

The feedback you collect during beta is only valuable when it gets into the right hands at the right time. Nobody wants to find out that a critical bug made it to release or, even worse, that the bug was discovered during beta but went unhandled. You can accelerate the triage of discovered defects by using beta tools that integrate with the bug-tracking tools used by your QA and Development teams. When eyes are on the issue quickly, you also have an opportunity to seek more information from the beta tester and regress possible fixes.

Feedback should also extend beyond defects. Survey responses, journals, forum posts, and other tester feedback can all have significant value for your support and marketing teams, which can result in lower support costs, fewer product returns, and better sales messaging. Essentially, the key to increasing ROI through feedback workflow is to focus not just on collecting data, but also on ensuring that it's actionable during testing.

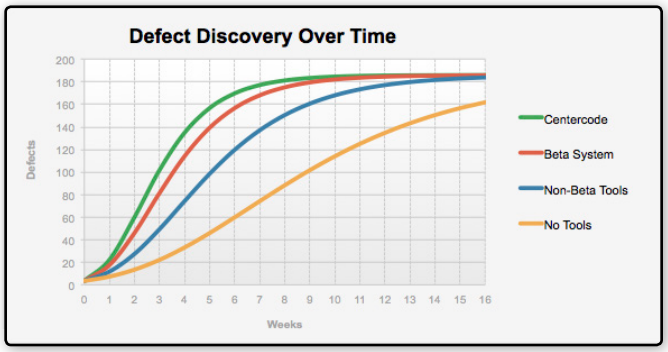
C. Beta Participation and Total Testers

Beta tester participation and the total number of beta testers are also major targets for ROI improvement. For many companies, beta participation rates are as low as 30%, and the 70% of testers that aren't participating are a significant drag on resources. Tester recruitment and selection, agreement execution, product distribution and retrieval, and participation management/support all scale with the number of testers. And for hardware beta tests, simply providing those testers with the product can be a large expense. Increasing beta participation to 80% or more, drastically reduces the number of testers required to achieve the same (or better) results. This cuts costs for beta management, as well as logistics and production.



D. Length of Beta Test Phase

Time is a luxury that companies often don't have, so getting more feedback out of a limited testing window is a major part of improved beta ROI. One key factor is to make sure you're maximizing active test weeks. If you have four weeks for beta, and two are spent finding and profiling beta testers, you're limiting the effectiveness of your beta. The Centercode beta test management platform is focused on getting better results from shorter beta tests, and half of that equation is ensuring shorter pre-test phases that let you launch a beta test in hours or days, instead of weeks.



If you have the luxury of time, then longer beta tests can often be very effective in increasing ROI. Our service client will often extend a beta test when feedback and tester interest are still going strong. Your beta team just needs to be organized and equipped with data that enable an informed decision about whether it's smart to go for a longer beta test. Otherwise, you're just increasing your beta management costs without a corresponding gain.

THANKS FOR READING

We hope you found our whitepaper worthwhile and that it will help you prove the value of your beta testing efforts throughout your organization. If you haven't yet, be sure to open the companion spreadsheet, which does a lot of the calculation work for you and offers baseline values to work from if you lack data. Whether you're building a business case for a single test or an entire beta program, this tool can speed up the process tremendously.

If you'd like to learn more about how Centercode and our beta test management platform can improve your own beta ROI (and your beta program in general), we'd be happy to schedule a demo and/or consultation. Please visit www.centercode.com/demo to request a demo, or www.centercode.com/plan to request a beta test consultation.

Centercode

Beta Test Management Solutions

Corporate Headquarters
23332 Mill Creek Dr. #206
Laguna Hills, CA 92653

Email: roi@centercode.com
Online: www.centercode.com
Twitter: @centercode

© Copyright 2013 Centercode

Special thanks to Adam Wright for the hundreds of hours that went into making this project a reality.

Readers are free to distribute this within their own organizations without modification.

All other company and product names may be trademarks of the respective companies with which they are associated.

Image courtesy of Flickr user zzzack.