Building a Better Beta

A Detailed How-to Guide on Running a Successful Beta Program

Saeed Khan

On Product Management.net

Contents

Introduction **3**
Origins of beta programs 3
What a beta is not 3

Beta program overview **6**

Phase I – Pre-Beta **7**
Defining beta objectives 7
Beta Requirements Document (BRD) 9
Beta prospecting 9
Prioritizing beta customers 11
Recruiting internal resources 11
Internal education 12
Educating beta sites 12
Providing test case scenarios 12
Beta software readiness 13

Phase II – Beta **16**
Launching beta 16
Working with beta sites 16
Recording and disseminating customer feedback 17
Providing technical support 18
Customer visits 19
Developing reference sites 19
Extending beta 20

Phase III – Post-Beta **22**
Final customer calls 22
Summarizing and addressing beta findings 23
Internal post-mortem sessions and publishing results 23

Conclusion **24**

Introduction

Is it just me, or is delivering a successful beta program one of the hardest things to do as a product manager? The difficulty arises because so much is out of the product manager's direct control and in the hands of the beta testers themselves. And if you can't get motivated beta customers to provide you with detailed feedback, you can significantly diminish or delay the revenue potential for your product.

Managing an effective beta program means doing what it takes to minimize the aspects that are out of the product manager's direct control and, like any other program that is executed, bring as much predictability to it as possible.

Think about it. A beta program involves having people spend their valuable time using your unreleased, and in many cases, buggy software. And on top of that, you usually have a fairly short period of time to gain valuable insights on what issues need to be addressed with the product before it is released.

Origins of beta programs

The terms "alpha" and "beta" were coined by IBM in the 1960s to label product development checkpoints. Alpha meant a unit, module or component test phase and beta was the initial systems test. These terms came from even earlier A, B and C tests for hardware. The A-test was a feasibility test that was done before conducting any design or development. The B-test was a demonstration or prototype showing that the engineering model functioned according to specifications. And finally the C-test (which maps to the current usage of the beta test), was performed on early samples of the product being created.

Software built today is quite complex, and whether it is called beta testing, external validation, early release or something else, it is almost universally accepted that software development is an iterative process and that user feedback is critical to developing great software products. And to maximize the value of the acquired feedback, a structured process must be in place to manage the beta program from end to end.

What a beta is not

Before I get into the details of what a beta program is and how to optimally manage it, it's important to understand what a beta program is not. It's not:

- a substitute for good quality assurance
- a substitute for proper design and usability testing

- a tool for the sales team to help close deals in their funnel
- something that should be confined to the product management or product development groups in a company

First, a beta program is not a substitute for good quality assurance. Sending out an early release to beta testers and counting on them to find bugs that should be fixed is a very bad objective for a beta program. In fact, if your development or QA teams disagree with the previous sentence, then they should realize that what they are saying is that something which is fundamentally their responsibility should be outsourced to a group of external people. The next logical step would be to outsource the entire development or QA function altogether.

Related to the previous point, a beta program is not a substitute for proper design and usability testing. There are two major reasons for this. The first is simply that something that is an intrinsic part of the software development process cannot be left to beta customers to resolve.

Efficient user interfaces need to be designed and implemented with an understanding of user needs. They must be based on easily understood cognitive models, support specific workflow patterns, minimize hidden or hard-to-access functionality, and in many cases, be adaptive or configurable to support a diverse pool of users. These things need thought, planning, and sufficient lead time to implement. And thus, if left until beta, almost guarantee that significant rewriting will be needed, possibly causing a delay in the product's release date.

Next, a beta program is not a tool for the sales team to help close deals in their funnel. I'm not saying that exposure to beta software can't help close a deal, but direct prospect participation in a beta program is a bad idea. Why not give a prospect early access to software to get them excited about what is coming down the pipeline? Well, first of all, you cannot sell beta software and there are easier ways to provide prospects information about what is coming in the next release. This can be achieved via on-site demonstrations, Webinars, or even white papers and presentations.

Additionally, simple software products notwithstanding, the cost of supporting a beta site and the risk of exposing the prospect to potentially buggy functionality (it is not fully tested and debugged, after all) may outweigh any advantages or benefits in accelerating the software sale.

Finally, a beta program is not something that should be confined to the product management or product development groups. If a software company is working on a major product release, and particularly if that product generates a significant portion of the company's revenue (i.e. it is not a marginal product), then people from across the

entire company must work together to set objectives for the beta program and ensure they are met or exceeded.

After all, the beta program is almost always the first external exposure of the upcoming software product. And if this product is expected to drive future revenues for the company, then the value of, and investment in, the program must be looked at from an overall company perspective. Virtually every customer-facing group in the company should be involved in the beta program. Whether the goal is related to technical enablement for groups such as Customer Support or Professional Services, identifying press and analyst references for Marketing, or accelerating time to revenue for Sales by developing early adopter sites; taking a broad, company-wide view of the beta will maximize the benefits of the beta program for the entire company.

Beta program overview

Beta planning and execution can be broken up into three phases, with each stage having multiple tasks and objectives. It's important to note that the following table provides a detailed set of tasks for a comprehensive beta program. Not every beta for every product will need all of these tasks. It's up to you to decide how much or little you need to follow for your beta program.

Phase I – Pre-Beta	Phase II – Beta	Phase III – Post-Beta
Defining beta objectives and the Beta Requirements Document	Launching beta	Final customer calls
Soliciting beta customers	Working with beta sites	Summarizing beta findings
Prioritizing beta customers	Recording and disseminating customer feedback	Addressing beta findings
Identifying internal resources and resource readiness	Providing technical support	Internal post-mortems
Educating internal teams and beta customers	Customer visits	Publish post-mortem results
Creating test case scenarios	Developing reference sites	

Phase I is the pre-Beta (or planning) phase.

Phase II is the actual testing (or execution) phase.

And Phase III is the Post-Beta (or analysis) phase.

The following sections provide details on each of these phases and the tasks in those phases.

Phase I – Pre-Beta

Defining beta objectives

As in most significant endeavors, planning and preparation are keys to success. With beta programs, success will depend heavily upon proper execution of activities before the beta testing actually begins. The first thing that must be done is to identify the goals of the beta. To do this, a number of questions must be thought through and answered.

What are the desired outcomes of beta? The main objective for beta is, of course, product feedback. But, what kind of feedback is needed? Are there specific functional areas of the product that require feedback? Is it focused on the user interface, performance, scalability, installation, upgrade, etc.? Are there certain configurations, platforms, use cases, or others scenarios that need to be tested? How many beta sites need to provide feedback for each of these configurations, platforms, use cases and scenarios?

For example, if the product is something like an application server, with new features related to scalability and transaction processing, as well as support for the latest versions of Linux®, you need to define how many beta participants you want testing these features, how many Linux beta testers you need, and also, as a control, how many beta sites you need testing on non-Linux platforms.

This helps you identify whether there are any issues specifically related to the new platform. You should also see if existing customers can do some head-to-head performance and scalability tests comparing your old version with the new one.

Beyond product feedback, it is typical in enterprise software companies to solicit some beta participants as reference sites for press quotes and analyst briefings. How many of these reference sites are needed and what is the profile of those sites? Do they need to be direct customers or can they be partners? Do they need to be large, name-brand enterprises or can they be smaller, less well-known companies? Do they need to be in particular geographies around the world? If so, how many are needed in each geographic region?

Other questions to answer when planning for beta include defining the beta timeframe.

When will it start, and when will it end?

Assuming you have software that needs to be installed at customer sites, will customers install it or will you send someone there to install it for them?[1]

How will you distribute patches or interim builds to customers during the beta program?

Do customers need training on the beta software? If yes, how will it be delivered?

When will it be delivered?

How will customers communicate their feedback to you during beta? How will they receive technical support?

Will beta customers be allowed to communicate with each other?

If yes, then how will you make that happen?

These are key questions that should be discussed internally and agreed upon in the early planning stages of the beta program.

In a beta I managed for a major software release, we discussed all of the issues above and decided the following:

- The beta would run for a period of three months, starting about seven months before the expected release date of the product.
- We would ship software to all customers on DVD media at the beginning of beta, but only provide patches on a one-off basis to those encountering serious problems while beta testing.
- For training, we would hold three 90-minute Webinars for the beta customers in the week after software was shipped. These Webinars would cover the installation process and all of the major functionality areas. We would also have live demos on these functional areas in the Webinars.
- We would set up weekly 30-minute calls with each beta site and assign a product manager and a program manager to each customer to manage communications and track their progress.
- Customers would use our standard customer support process to access technical assistance during beta and would use the weekly call to discuss higher-level issues such as specific product feedback.

---

[1] For on-premise software products, virtualization or cloud services (e.g. Amazon EC2) can significantly help with deployment of beta software. i.e. providing preconfigured VMs or pre-loaded cloud-based servers that customers can access eliminates the need for them to allocate hardware or configure the software.

Beta Requirements Document (BRD)

As you can see, there are many questions regarding the beta that need to be answered in the planning stage.

All of this information should be collected and documented in what I call a Beta Requirements Document (BRD). The BRD is a single point of reference for all aspects of the beta. It contains information on the key internal stakeholders for the beta (i.e. from Product Management, Product Marketing, Customer Support, etc.), their responsibilities, key milestones, and process descriptions about the beta (i.e. how many customers are expected in the beta, their proposed profile, the beta registration and approval process, how software will be distributed, how support will be provided, how feedback will be collected and recorded, etc.).

By creating and publishing the BRD, anyone within the company who needs an understanding of the beta program can read the relevant portions. It also becomes a reference point against which you can measure success after you have completed the beta.

Beta prospecting

The BRD should define the ideal profile of the participants you want to have in the beta. How many total beta participants are required? What mix of new and experienced users do you want? What mix of direct customers and partners are needed, etc.? But getting the right number and mix of participants to commit to beta is another story.

Remember, participating in a beta program means that people are going to spend their time on your unreleased software. Given peoples' busy work schedules, finding a significant number to actively participate during the timeframe of your beta is a challenge.

Look at prospecting in the same way you look at sales: build a funnel of qualified prospects that is some multiple (e.g. two or three times) of the total number of beta participants you need, and then work to close as many of them as possible. And of course, to build this funnel, you will need to contact an even larger multiple of people.

As an example, in my beta program, we identified six major functional areas where we wanted beta feedback. We also decided that our goal was to have between three to five active beta participants per functional area. By "active" I mean people who would regularly using the beta software and providing meaningful and detailed functionality feedback.

So, three to five active participants per area translated to 18 to 30 active beta participants overall. But ensuring that we met this number meant recruiting at least twice as many into the beta program.

There are always sites that initially commit to beta but then drop out or do not participate as actively as planned, so ensure that you have significantly more than the target number committed to beta. Assuming roughly a 50% drop off between those who commit to beta and those who actually participate, we decided we would need at least 40 to 50 fully qualified and approved beta participants at the beginning of the program.

We did a lot of prospecting to acquire these qualified sites. Over 500 customers and partners were contacted. We worked closely with our sales and marketing teams to identify prospects. We also canvassed customers we had visited in early requirements gathering trips for functionality in this release.

Part of our qualification process included understanding why the person or people were interested in participating in the beta. Was there a critical business problem that needed to be solved or did they just want to "kick the tires" of the new release? We also found partners, specifically VARs, SIs, partner ISVs, and even independent consultants to be particularly good beta candidates. Most had a clear business reason for wanting access to the beta and had staff that could be dedicated to the beta and work with us to provide very detailed product feedback.

With direct customers, the story was a little different. While we identified many well-intentioned direct customers, spending time on beta meant an additional task that had to be done on top of their full-time jobs. Thus, it was something that many of our beta contacts were not keen to tackle. In those cases, we worked with our account managers to escalate the benefits of the beta program higher up at the customer site and tried to get a management sponsor for participating in the beta. Overall, this was a very laborious process, and took about two months to complete.

At that company, Product Management recruited the beta sites, but product managers I have spoken with in other companies say that they have recruiting teams in their organizations who call and solicit beta customers, though in those cases, it is still Product Management's task to define the beta customer profiles and recruitment criteria.

Keep in mind that recruiting is the most critical aspect of the beta program. Get it wrong—the wrong sites, the wrong mix of sites, or too few sites—and you are unlikely to meet any of the beta program's downstream goals. And while getting it right does not guarantee success, it does significantly increase the probability in reaching those goals.

Prioritizing beta customers

Not all beta customers are created equal. During the prospecting process, it will become evident that some customers are more likely to provide good feedback than others. As mentioned earlier, those with clear business needs addressed by the beta software, as well as partner companies, typically have the incentive to invest time and resources into your beta program. In addition to these companies, there may be some beta participants who are considered marquee customers and are potential beta reference sites. In short, a subset of the beta participants will warrant extra attention and support during the beta program as they are more likely than others to help you reach your targets.

Identify these marquee companies early. Make all internal parties in your company who are involved in the beta program aware that these firms will be monitored closely, and if needed, given additional help and support to ensure they are successful in the beta. Assign named technical resources to these participants to better ensure their beta program success.

The product management team should work closely with these customers, visiting them or engaging in additional contact to help resolve issues that arise.

You may not be able to accurately identify all marquee sites until the beta is actually in progress. There are many things that can happen between the time a customer commits to participating in the beta and when the beta actually begins. In some cases, what looked like a promising site during recruitment turns out to be a dud once testing begins. And the opposite is also true. What sounds like a sleepy beta prospect turns into a very active and engaging beta site that far exceeds the original objectives. There is no harm in updating the marquee list once beta starts, and in fact, it may become a necessity as the beta progresses.

Recruiting internal resources

As beta recruiting is taking place, you need to ensure internal resources are identified and will be prepared for beta. Who will provide technical support to the beta sites? Is it individuals from the technical support organization, from professional services, sales consulting, or could it even be from your development and quality assurance teams? It varies across companies, typically depending on the nature of the software and the size of the company. For smaller companies, beta support may be delivered by Engineering, whereas in larger companies, the Customer Support team may handle it.

Internal education

Once you have defined the internal team members, ensure appropriate individuals—particularly those who will provide the technical support for the beta—are properly educated on the beta software.

Depending on the complexity of the product, this could be as simple as giving the internal groups access to the software and having them educate themselves via the documentation, or it could range all the way up to delivering formal training classes ahead of the release of beta software to customers.

The objective ensures that your internal teams are at least one step ahead of customers, so that as customers start installing and using the software, the internal teams are ready to respond to any problems.

Educating beta sites

Beta customers typically need some form of education or training before using the beta software. If the software is a consumer-oriented product, then ideally, little or no training should be required. Worst case, the customer should be able to read or view a short online tutorial to get them going. Conversely, complex business or enterprise software may require extensive training.

As mentioned earlier, we held three 90-minute Webinars for our beta customers. Virtually all beta customers attended all three Webinars and we received positive feedback from the attendees. While that made us feel good, we also knew that Webinars would only partially prepare customers to use the product and we would have to help them by explaining functionality during the beta itself. We also provided documentation with the beta, but that too was in a "beta" state, and while most sections were reasonably complete, more detail could and would be added later in the development cycle.

Providing test case scenarios

Without some guidance, a lot of beta feedback can be vague and generic. Statements like, "I tried this, this and that, and they all seemed to work OK," or "It seems to work as I would expect, but I noticed the buttons in the dialog box are not properly aligned.", are not uncommon when testers are left to their own devices.

To help guide the testing done at each site, provide the beta customers with some structure. For example, define beta test scenarios for each area of functionality that needs to be tested. The purpose of these scenarios is to provide a clear framework to the customer in which to test the software. It is not to provide a step-by-step set of actions that they should perform.

The scenarios should focus on typical use cases you expect customers to perform once the software is released. The scenarios can also focus on specific aspects of functionality that need testing in customer environments. Do not expect many beta customers to be able to perform in-depth stress testing of your software. This can be rather time consuming to prepare for and execute, and most beta customers will not have the time and resources to do this level of testing.

The scenarios should be made available at the beginning of Phase II and can be introduced during the beta Webinars or in the first customer calls once beta has begun.

Beta software readiness

Another important task in preparing for beta is to track the readiness of the beta software. There are two major questions to answer:

How can you measure whether the software is ready to ship when the beta begins?

What do you do if you determine that it is not ready?

It comes down to whether you want to be date-driven, quality-driven, or in most cases, an acceptable mix of the two.

Determining shipping readiness means measuring the quality of the software and deciding if it meets preset, agreed-upon criteria. If it meets the criteria, it is ready to ship. If not, more work needs to be done on it. Software quality can be an elusive concept, so how does one measure it?

The first thing that comes to mind is the number of open bugs and their severity. A lot of discussion will take place in Development or QA meetings about the number of critical or showstopper bugs that are in the current release. Many times statements are made that sound something like, "We cannot release the beta until all critical bugs are fixed and there are less than X number of severe bugs.", where X is some arbitrary, but reasonable sounding number. The problem with a statement like this is that, in reality, it is very difficult to enforce.

First, as the target beta date approaches, and if the number of critical and severe bugs remains high, the definitions of both critical and severe will magically loosen. What was once a critical bug two months prior to beta becomes a severe bug two weeks before beta. And what was once a severe bug, may become a moderate bug.

Second, what if all but "a few" critical bugs can be fixed by the beta date, will you still ship the beta? What if, instead of X severe bugs, there are 10% more than X severe bugs, will you ship it? You can always issue a patch or update of the beta software later, can't you?

Third, the statements about number of bugs are one-dimensional. The statements treat all critical bugs the same and all severe bugs the same. In reality, different functional areas need to be tracked separately with metrics for each area. In some cases, there may be lower tolerance levels acceptable for bugs, in other cases, it may be higher.

Overall, things like usability, performance, installation, and upgrade path need to be measured and tracked separately with their own criteria and requirements.

For example, if your software has an installer, the installer *must* be bug free (unless you will provide pre-installed VMs or hosted servers). If the installer is buggy, not only will the initial user experience be negative, but in cases where the user encounters a bug, the user will either have to call Support for help, or perhaps worse, abandon the install altogether. If you don't track this as a separate item and only look at aggregate bug numbers of different severities, then installation bugs will come back to haunt you.

Usability issues are another facet of applications that must be tracked separately. Many GUIs do not pass through rigorous design, review and implementation processes. They may be implemented by engineers, who may not be design experts, and then modified later as issues are raised. The GUI is the face of the product, and the best server or back-end software can only be hampered by a difficult GUI.

With respect to GUIs and beta programs, remember that:

- The beta is usually the first exposure the public has to the software and they will form opinions of the software based on the beta. A significant part of that opinion will be based on their emotional experience with the GUI.
- Many of these same people will be candidates for marketing reference sites. If their emotional experience is negative, they will likely not want to be associated with the product, even if you tell them that everything will be "fixed" when the software is released.
- Usability reviews should be performed on the pre-beta software and a commitment made to address the most important issues before the beta. The completion of those issues becomes one metric used to measure release readiness.

All of the readiness criteria should be converted to a number of metrics that are tracked and reviewed via a management dashboard. This dashboard shows high-level status for each category (e.g. via red, yellow, and green indicators) with the ability to drill down into details where needed. If available, you can use a dedicated reporting or dashboarding tool. But if not, then virtually any application, such as a spreadsheet, presentation tool or word processor will do. Regardless of how it is implemented, there are a number of benefits to this approach.

A management dashboard provides a top-down view of the state of product quality. A lot of red on the dashboard a few weeks before beta is set to launch may give you early warning that you will need to slip the release date or that additional focus or resources are needed to meet your existing dates.

If most areas show green, but a few are persistently red or yellow, it may indicate issues with the teams working on those areas of functionality, and specific action can be taken to help those team members. Another obvious benefit of such a dashboard is that it gives upper management clear visibility into the status of the software and the ability to make better and quicker decisions to reallocate resources to where they may be needed.

Phase II – Beta

Launching beta

After all of this preparation, you are now ready to deliver the software to customers and begin the actual beta testing. In most cases, customers should be able to install the software on their own. If the installation is complex, you can either send someone onsite to help install the software, or provide preinstalled software on a server or desktop machine. Keep in mind that while it may be possible to provide extra installation assistance during beta, this is not going to be the case when the software is generally available. So, if the beta requires extra effort to install and configure, make sure you know why that is the case, and have a very clear and firm plan to address it before final software is released.

Working with beta sites

The major aim of the beta is to collect feedback from the beta sites. Despite any assurances to the contrary from beta sites, most sites will need a lot of prodding and assistance during the beta program if you want their substantive feedback.

Based on my own beta experiences, combined with some anecdotal data from other enterprise software product managers, roughly half of all beta sites (50%) will end up installing and using the software in some way, and roughly half of those (25%) will provide meaningful feedback on at least one area of functionality.

Thus, if you start with 50 registered beta sites, approximately 20 to 30 will actually install the software and provide some level of useful feedback, with only about 10 to 15 giving detailed and very meaningful feedback. Of course every situation is different, but these ratios have held true for me across several beta programs.

Regardless of the number of active beta sites, there are many ways to collect feedback. Depending on your goals, you can use the following techniques:

- Weekly calls with individual participants
- Regular conference calls with multiple participants
- Beta participant blog and/or discussion forums
- As needed email and phone conversations

Weekly calls with individual participants work very well. Although it can be time consuming to hold weekly meetings with each beta site, customers will commit to a

weekly 30-minute call, and in some cases a 60-minute call, to discuss their issues and feedback. The benefit of the weekly call is that the beta customer can block off a fixed amount of time each week and knows they will get your undivided attention for that period of time.

If they have problems with the beta software, the weekly call creates an opportunity to discuss the issues and get the customer back on track. If they are making progress with the beta, this weekly call gives you the opportunity to discuss their feedback in great detail and guide them further along in the directions you want them to go.

In addition to weekly calls with individual beta customers, you can set up regular, perhaps biweekly, conference calls with multiple customers. There is a risk of these calls devolving into a gripe session—it only takes one unhappy beta customer to get a group started—but if managed well, listening to customers discuss their issues with one another can lead to some valuable insights.

Another mechanism for communicating with beta customers is to provide a blog or web-based discussion forum to enable them to ask and answer questions, post findings or simply discuss relevant issues with one another.

The benefit of providing an electronic forum is that the information is automatically persisted for you, and if supported by the software, customers can upload files, images, reports or any other documents that are related to the beta and their findings. These web-based forums can also be used as customer self-help tools, enabling beta customers to ask and answer questions amongst themselves.

Last and—in my opinion—least, let the customer contact you via email or phone on their own schedule and not set up formal weekly or biweekly calls. While a dedicated beta customer may be quite proactive and eloquent if left on their own, in my experience, leaving it up to the beta customer to contact you when they can will lead to little, if any, meaningful feedback.

Recording and disseminating customer feedback

Collecting customer feedback is one thing, but recording and disseminating it within your organization is something completely different. Remember that different internal groups require various levels of detail of the beta feedback.

Obviously the most detailed information is in the form of the raw notes and findings taken from weekly customer calls. This raw feedback should be posted in an internal forum, such as on a wiki or portal, so that all parties can access it easily. The main consumers of this level of information will most likely be engineers and product managers. Raw feedback contains various types of information about customer issues,

positive and negative comments on various parts of the software, access to technical support, plans, expectations, etc.

The information collected from these calls should be sliced and diced, and presented in multiple ways so that it can be efficiently utilized. Information can be available in any (or all) of the following ways:

- Weekly call notes by customer (i.e. all calls for individual customers)
- Customer call notes by week (i.e. all calls for a given week)
- Detailed feedback by area of functionality
- High-level summary of beta findings
- Overall beta customer status by week

There may be other ways to present the data, but those listed above will give most interested parties what they need. Account managers will be very interested in the weekly progress of their accounts. Product managers and development managers will focus on the detailed feedback by area of functionality. Overall beta customer status by week is interesting to track.

As customers receive software or begin their beta participation, it is important to track a high-level status for them each week. This status should be based on clear indicators of their progress during the course of beta. A set of statuses, including registered, received software, installed software, testing in progress, blocked, and completed beta, applied to each beta customer each week can be a valuable tool in tracking issues and progress during the beta. These types of simple metrics can be used to create dashboards or indicators so senior management can get a clear picture over time of the progress and status of the beta program.

For example, if early in the beta program a significant number of customers are blocked by installation or configuration issues, this will be clearly visible in the dashboard. Corrective action can be taken, but if the issue persists, decisions can be made to either resend a newer version that installs correctly, provide additional on-site help if warranted, or even consider extending the beta to make up for lost time.

Providing technical support

Most software companies have well-defined policies and procedures for delivering customer support. Typically each customer has some form of maintenance agreement with the vendor. But during beta programs, where even small issues can block customer progress, there may need to be a simplified and expedited mechanism for customers to log issues that need to be resolved. I have heard many customers request this during beta

programs. This could mean that the normal process to log and troubleshoot cases is too cumbersome, and while that may be true, it can also indicate that customers are not willing to invest as much time to resolve beta issues as they are regular software issues.

An expedited process should be set up for beta. It can be as simple as a dedicated email address for customers to use to indicate a problem. If you prefer a little more sophistication, it is very simple to set up a structured web form that customers can use to submit issues.

Another idea is to set up a shared discussion board for beta customers so they can log issues. Resolutions to common problems can also be posted on the discussion board. Sometimes, a beta customer who logs one problem may be able to offer a solution to another. This way, not only can some of the load be taken off of the customer support team, but beta customers can achieve a level of self-service in resolving problems.

Customer visits

As every product manager knows, there is nothing like a visit to a customer site to get a deeper understanding of customer issues. Additionally, customers often look forward to visits from product managers and will put aside several hours of their time, if needed, to discuss their concerns. So what better time to visit customers and discuss issues and concerns than during a beta program?

Part of the beta planning process should include budget and goals for customer visits. The primary goal of the visits is to get more detailed information about the beta experience of the site, but do not limit it to that. The visit is a great opportunity to discuss future plans for your product in their organization. You can also uncover new projects or initiatives where your product can be used and even meet other people in the organization who can influence the use of your product in the company.

Be very careful to keep the tone of the visit as a research visit and not turn it into a sales or prospecting call. As a product manager, you have a lot of credibility and influence with the customer, so don't jeopardize that by focusing on sales objectives instead of the customer needs.

Developing reference sites

In my experience, there is no direct relationship between someone having a positive beta experience and their ability to become a reference site. Certainly a positive experience improves the chances of someone becoming a reference, but there are numerous factors that can prevent that.

Many companies have strict policies about their employees speaking publicly on behalf of a vendor. For those companies that allow their employees to speak publicly, there are usually strict guidelines as to what they can say and to whom they can say it. Those who do speak usually do so because they have a very good personal relationship with individuals at your company, and do not necessarily need to be actively involved in a full-blown beta program to assess the product and determine their opinion. Perhaps a live demo or a customer day at their site can be sufficient to get them acquainted and comfortable enough with your product to speak about it.

The point is that you should not depend *solely* on the beta program to develop press and analyst references for your product launch. Certainly, some beta sites can turn into references if your beta program progresses well, and those who have good beta experiences also have the right profile, authority and permission to speak on behalf of their company about your product.

But, that leaves far too much to chance. What if very few beta sites have a good experience? What if those sites that do have a good experience with the beta do not fit the profile of type of reference site you want? What if those who do fit the profile and have a good experience do not have the permission to speak publicly?

To mitigate these risks, you need to have a separate but parallel effort to identify potential reference sites and work with them to get them to a point that they will become references. This effort is typically performed by Marketing or Product Marketing and not by Product Management.

As mentioned earlier, the reference prospects may already have a personal relationship with individuals in your company and may not need full beta exposure to gain the product knowledge they need in order to speak about it. If some of these potential reference sites do require beta participation before becoming references, then let them participate. But once they enter the beta, all interaction with them should be handled with the full knowledge that they are a potential reference site, and not a site that will necessarily provide detailed feedback on features and functionality.

Extending beta

Depending on how your beta program progresses, you may want or need to extend the time period over which you hold the beta. Sometimes customers start late in the beta cycle and want additional time to provide feedback or the beta program identifies a number of issues in the product, and you realize that those issues are important enough that more product feedback is needed on the product. Regardless of the reason, it is important to know that this is a possibility and to prepare for it in advance.

If you embed a time-bound license with your beta software, make sure that you can update that license if you extend the beta. If you cannot extend it after the fact, build additional time into the original license so that if you extend the beta, those customers who do continue to use the software for the extended period can do so without interruption.

Also, make sure that the internal resources such as Technical Support and Professional Services who will be assisting customers in the beta are available beyond the initial beta period, if needed. Their managers may not be aware of the potential extended timeframe and may have other plans for them once the regular beta period ends.

Phase III – Post-Beta

Final customer calls

During Phase II of the beta, there is regular customer contact focusing on progress with the beta, with the supplied use cases and any issues that the customer faces with the product. This is typically tactical information focused on the current week or stage of beta customer participation. To close out the beta program with customers, you should hold a final wrap up call to discuss higher-level issues and obtain feedback on the overall beta program. This information is extremely valuable in assessing overall beta success and to identify the do's and don'ts for future beta programs. The following is a short list of questions I have asked customers at the close of beta programs. Note that these are discussion points, and not survey questions, and it is up to you to drill down with the beta customer to draw out additional details, where needed.

- Did the training provide sufficient information to help you understand the capabilities of the beta software?
- Were the weekly conference calls helpful to you?
    - What was helpful about them?
    - What was not helpful, and what could we improve?
- What could be done to make the time on the calls more productive?
- Did you find the process of requesting support intuitive and straightforward?
- Did the technical support process meet your needs?
- How could support be improved for future beta programs?
- Did you have adequate hardware to test out the beta?
- Were you able to easily get up and running with the beta software?
    - What could be done to make the process easier?
- For each area of functionality tested, what were the standout positives and negatives?
    - What changes/enhancements are critical to the initial release of the product?
- Do you plan to upgrade to the new release? If so, when? If not, why not?

Given the detail of some of these questions, you should schedule at least one hour for these calls and ideally send the questions in advance to the beta customers. Getting the questions in advance will let them collect detailed and accurate answers to these questions and, if needed, poll their coworkers who also participated in the beta program.

Summarizing and addressing beta findings

Once the beta is over and the final customer calls are wrapped up, all the findings need to be consolidated and summarized into product and process categories.

The product feedback from the beta must be converted into specific requirements that need to be addressed. The critical requirements can be added to your overall release criteria for the product. That is, the software cannot be released to market without addressing these critical needs. The remaining requirements can be scheduled for future feature or maintenance releases, and their priority can be assessed in the context of other requirements for those releases.

A report should be created that summarizes the findings and any other notable details gleaned from the beta. This report is intended for Sr. Management or other groups who are interested in the beta but don't need the detailed product requirements derived from the beta feedback.

Internal post-mortem sessions and publishing results

The process findings also need to be understood and implemented. But unlike the product findings that will come almost exclusively from beta customers, the process feedback will have both an external as well as internal component.

To obtain the internal beta process feedback, beta post-mortem or review sessions need to be held. These sessions should be led by Product Management (or Program Management) and be held with individual teams such as Technical Support, Marketing, Quality Assurance, and Development.

The reason to hold them with separate teams and not in one large session is so the discussion can remain focused and can drill into detail on the issues for each group.

Once the individual sessions are held, the findings from all the groups can be combined and presented back in a joint meeting to the internal teams. This final meeting helps bring all groups together and everyone can collectively discuss what should and should not be done in future beta programs.

Conclusion

Executing a successful beta program requires approaching it in a very analytic manner. You must focus on minimizing the inherent uncertainty in collecting feedback on a pre-release product from very busy customers and partners. No one will argue that beta programs are a key element of the software development process, but planning for and executing a successful beta program is a difficult task. By taking a rigorous and methodical approach, and working to make each individual beta customer successful, you increase the likelihood of meeting your goals, delivering necessary market feedback to your organization, and significantly enhancing the chances of success for your product.