How To Write a Good PRD

Martin Cagan

Silicon Valley Product Group

SVPG

Silicon Valley Product Group

HOW TO WRITE A GOOD PRD

Martin Cagan, Silicon Valley Product Group

OVERVIEW

The PRD describes the product your company will build. It drives the efforts of the entire product team and the company's sales, marketing and customer support efforts. It's hard to come up with a more important, higher leverage piece of work for a company.

The purpose of the product requirements document (PRD)[1] or product spec is to clearly and unambiguously articulate the product's purpose, features, functionality, and behavior. The product team will use this specification to actually build and test the product, so it needs to be complete enough to provide them the information they need to do their jobs.

If the PRD is done well, it still might not be a successful product, but it is certain that if the PRD is not done well, it is nearly impossible for a good product to result.[2]

The PRD versus the MRD

We draw a distinction here between the product requirements and the market requirements (often referred to as the "MRD"). Put very simply, the market requirements describe the opportunity or the market need, and the product requirements describe a product that addresses that opportunity or need.

The PRD versus the Product Strategy and Roadmap

The product strategy describes a vision, typically between two and five years out, of where you want the product to go, and the product roadmap describes the various steps to get there. The PRD describes a particular product release along that path.

[1] Note: Different companies and industries use different terms or acronyms to refer to the product requirements. Similarly, some use a single document to capture market, product and interface requirements, and others use a series of documents. For our purposes, we simply refer to the "product requirements" and use the acronym PRD. We intentionally do not refer to MRD which we consider distinct.
[2] This paper is based in part on the notes and insights of Ben Horowitz, President and CEO of Opsware.

TEN STEPS TO A GOOD PRD

The purpose of this paper is to describe a proven, repeatable process to create a good PRD. The ten steps described here are not easy, but they can help you produce a strong PRD. The amount of time this process takes depends greatly on the size and complexity of your product, and how prepared you are in terms of the knowledge and skills required.

Step 1: Do Your Homework

Your goal with the PRD is to come up with a compelling product. In order to do this, you must do your homework. This means studying your customers, your competitors, and your team's capabilities, including available technologies.

This begins with customers, users, competitors, industry analysts, your product team, your sales force, marketing, company executives, other employees – anyone that has insight into the problem and possible solutions.

There is much more to the process of preparing, and this is discussed in detail in the paper "*Behind Every Great Product*" also available from the Silicon Valley Product Group.

Realize also that a significant factor in your ability to convince the team of the eventual success of the product is the degree of confidence you project, and you will be more confident and more convincing if you've done your homework well.

Step 2: Define the Product's Purpose

Every good product starts with a need that it is trying to fill. You must have a clear understanding of that need, and how your product addresses that need.

It is essential that the product manager establish a very clear, concise value proposition that lets her easily communicate to everyone – the product team members, company executives, customers, the sales force – what the point of this product really is.

While this sounds obvious, few products have such a clear value proposition. Consider the "elevator pitch" test. If you had a chance to ride the elevator with the CEO of your company, and she asked what the point of your product was, could you answer that question before the ride is up? If not, you have work to do.

It may be that the product does not have focus; it may be trying to do so many things that nothing clearly stands out. It may be that what you think is the big thing is just not resonating with customers. It could be that your product is trying to solve a non-problem; maybe you have a technology that you're still trying to find an application for.

The value proposition should also make clear how this product helps deliver on the product strategy. Note that you do not need to talk about every little feature; in terms of a clear value proposition, less is truly more.

The product requirements need to specify exactly what the objectives of this specific product release is, and how they will be measured. The objectives should also be prioritized.

For example, your objectives may be: 1) ease of use, 2) retail price under $100, and 3) compatibility with previous release. You could then go on to elaborate on how you will measure these objectives. For items like a specific retail price, it is straight-forward. For items like "ease of use," you need to be specific as to what level of usability the product requires. This is typically defined in terms of the target user. Usability engineers can rate the usability of your product for a given type of user. They can also rate the severity of usability issues, and you may specify that there will be no major usability issues.

The key is to be very clear to everyone involved just what success looks like, and to provide the guidance that the product team needs in order to make the necessary trade-offs in design and implementation.

Step 3: Define the User Profiles, Goals and Tasks

Now that you understand what problem you want to solve, the next step begins with an in-depth understanding of the target users and customer. During this step, it is important to work very closely with your product designer.

User Profiles

By this stage, the product manager will have hopefully met with many target customers, and have spent considerable time with first-hand observations and discussions. Now we need to classify the types of users and customers, and to determine who the primary users are.

For example, if you're building an Internet auction service such as eBay, you know that you have both buyers and sellers, and for each of those you have lowvolume/occasional users and high-volume/frequent users. It is not hard to imagine that there are several other less prevalent types of users as well, such as the corporate purchasing agents that may buy items for their company.

What the product manager and designer need to do is first identify the most important constituencies, and then try to characterize them in considerable detail, so that you can use these profiles to guide you in the design of the product. These profiles are sometimes called "personas" and while they should be fictitious, they should be as representative, realistic and plausible as you can make them. The idea is to come up with an archetype which captures the essence of this type of customer.

Here's an example:

> Leon the Power Seller is a 46 year old male that lives in Fresno and runs a small motorcycle parts business. While he does maintain a small shop, almost all of his sales come from eBay, where he sells on average 400 items per month. He sells a wide range of motorcycle related items, but his most popular items are saddle bags for Harley Davidson's. He owns two big Harley's himself, and he also drives a 1993 Toyota Pickup. Leon is married and has two teenage sons.
>
> Leon bought his computer just so that he could use eBay, and seldom uses anything except eBay and e-mail. Leon has been selling on eBay for three years now, and has learned what he needed to in order to effectively sell. He has a feedback rating of over 5000, which he takes great pride in. When eBay changes things on the site, especially the selling process, it can be very aggravating for him to learn the differences and change his procedures.
>
> Leon has well-established routines where he lists items to sell on Monday, and most of his auctions end by Friday, and he tries to get the shipments out within a few hours of receiving the payments.

Hopefully this description helps you get to know Leon and understand where he is coming from. As we consider new features, we can ask ourselves what Leon's response to the feature would likely be, and what we would need to do in order to get him successfully utilizing it.

Narrowing the set of profiles down to just the key ones is essential. Trying to please everyone is futile, and typically ends up pleasing no one, so it is important to try and come up with the few most important and prevalent profiles. Similarly, if you do not try and precisely characterize your target user, you will have this abstract notion and find it difficult to understand how your customers will react. You'll tend to assume that your customer is more like you than he really is.

User Goals

Once we have identified and characterized our main types of users, we then need to identify what each user's main goals or objectives are in using this product. This may sound simple, but it can often be challenging to untangle the underlying problem to be solved, when all around you people are telling you essentially the solution they think they want.

Everyone from the CEO to sales reps to engineers to customers are all too happy to "help" you come up with the solution. They'll tell you that you need to add a button or a shortcut in a particular place, or add a specific feature because a competitor has it, or change the colors because they don't like it.

The problem with jumping right to the solution is that there are often much better ways of solving a problem that the product manager and designer can come up with, if they are given the time and the freedom to come up with alternative solutions.

The best solution hinges on having a clear understanding of just what problem needs to be solved. The objectives may be different for each user profile, so it is important to be able to look at the objectives in terms of the profile they relate to. It is very possible that the feature being requested addresses a problem that is not one of the objectives of the primary user profiles.

Tasks

With the user profiles and their associated goals in hand, we can now move on to designing the tasks that help these people accomplish their goals. This is the heart of the product specification process, and is the place where creativity and innovation are to be encouraged and facilitated as much as possible.

Many outstanding products simply solve an existing problem in a new and better way. Sometimes this comes from the application of new technology, but mostly it comes from an insight that leads to a new approach.

For example, TiVo took the old problem of recording TV shows and came up with an entirely new approach that let customers accomplish their objectives much more easily, and established an entirely new category of electronics device.

Notice that we have talked about goals and tasks, but that we have not mentioned features. This is because features should be in support of required tasks that map to customer objectives. You will often find that many features either map to very low priority goals, or are simply extra functionality.

There are good reasons to eliminate any functionality you can in the name of making the required functionality more accessible. Ironically, the fewer features you have, the more powerful your product is often perceived. This is because the less features you have, the more features that your customers will actually discover and use, and the more they can successfully use, the more powerful they will perceive your product. The reason this is so counterintuitive is that most of us are not anything like our target customers, and we are willing to spend far more time exploring features and tolerating complexity than customers not in our industry.

Step 4: Define your Product Principles

By now you should be formulating some detailed ideas on your requirements and the user experience. You will, however, still be faced with countless decisions and trade-offs, and it is important that you have some criteria in which to make the best decisions for your product.

In most teams, each product team member has some ideas in their head in terms of good principles for the product, but rarely do two people have the same ideas, and these differences can lead to unpleasant surprises later.

It is very valuable to try and identify a set of product principles that will guide the entire team throughout the project. These principles will be specific to your domain and your particular project.

As an example, consider TiVo. When the team began on the product effort, the following product principles were established and shared across the team:

- It's entertainment, stupid
- It's TV, stupid
- It's video, damnit
- Everything is smooth and gentle
- No modality or deep hierarchy
- Respect the viewer's privacy
- It's a robust appliance, like a TV

These principles impacted the product definition greatly, and in many cases made for a much more difficult implementation, but arguably they are the source of the product's success.

Similarly, at eBay the mantra was: 1) easy to use; 2) safe; and 3) fun.

You can see how product objectives like these can provide a constant compass for the entire product team in the many decisions they must face during the course of the project.

Step 5: Prototype and Test the Product Concept

This is the stage where you actually come up with your product ideas. This is where you want to be as creative and innovative as possible.

One of the biggest and most common mistakes product teams make is to have far more confidence in their product specifications than they should, and they move forward and think they'll adjust the product, if necessary, once they get beta feedback. But of course beta is not the time for major changes, and hence no wonder so many first product releases are so far from the mark.

For most products, however, you can do a very significant amount of product validation testing at this stage, using various forms of prototypes.

First, it's important to consider carefully each of the three forms of testing you will likely need to do:

Feasibility Testing

One immediate question is whether or not the product is going to be possible to build. Your engineers and architects should be very involved in investigating technologies and exploring possible approaches. Some paths will be dead-ends, but hopefully others will prove viable.

What is most important is that if there are obstacles the engineering team considers insurmountable in this product's timeframe, it is important to know this now rather than discovering this much later in the process.

Usability Testing

Your designers (product designers) will be working very closely with you to come up with ways of presenting the functionality in the product so that the different types of users can figure out how to actually use the product.

Usability testing will often uncover missing product requirements, and also, if done well, identify product requirements that are not as necessary as originally thought. You should plan on multiple iterations before you come up with a successful user experience.

The purpose of the usability prototype is to have something to test on real people, and usability testing is the art and science of getting quality feedback on your product from your target customers. Certainly the product manager and designer will use the prototype and learn a great deal from it, but there is no substitute for putting the prototype in front of actual people from the target customer base.

Product Concept Testing

Finally, it is not enough to know that your product is feasible to build and will be usable, but what really matters is whether or not your product is something users will want to buy – i.e. how much do they like and value what you're doing? This testing can typically be combined with the usability testing.

For a few small product efforts, simply working your ideas out on paper is sufficient. But for most products, with complex user interactions or new uses of technology, a prototype of some form is absolutely critical in order to assess whether or not the product will meet the product objectives.

The prototype may be a physical device, or it may be a quickly assembled version of a software product. The key is that it needs to be realistic enough that you can test the prototype on actual target customers and they can give you quality feedback.

In the past, there were two major obstacles to prototyping. The lack of good prototyping tools meant that it could take a long time to actually construct the prototype. Another problem was in unaware management not understanding the difference between the implementation of a prototype and the real product, and the teams would get pressured to use the prototype as the basis for the final product, with predictable bad results in the implementation.

Today, there are outstanding prototyping tools that can let engineers or designers rapidly create very functional prototypes that can effectively emulate the future product, to the degree necessary, and form the basis of realistic user testing.

Moreover, most managers today understand that building a simulation and building the actual product are very different – akin to building a scale model of a house, and building the actual home.

It is impossible to emphasize enough how valuable it is to validate your product concept before you go and actually build the product. Once the real engineering begins it becomes very difficult to make significant changes and the costs of the changes rises dramatically.

Step 6: Identify and Question Your Assumptions

Once you think you understand the problem you want to solve, now is the time to start identifying and questioning your assumptions. It is very easy to make assumptions and not even be aware of them. Make sure you don't specify a candle in the PRD and prevent yourself from getting a light bulb.

Astronomy was originally defined as the study of how the sun and the planets revolve around the earth. Its very definition had an assumption that prevented anyone from getting the right answer.

Step 7: Write It Down

You'll need to get all this down of course. Most PRD's are Word documents, but some are Wiki sites, some are PowerPoint decks, and some live on whiteboards. The media and format are far less important than the content. Although what is critical is that the PRD be something that the entire team can easily access, it won't get lost, and that the PRD needs to be in a form that can be updated throughout the project.

Remember that a conversation is between two people. The PRD communicates to the entire product team.

It is also important to keep in mind that as important as the PRD is, what really matters is the product that gets shipped. Nothing is more important than that. So don't worry about how pretty your PRD looks or how thick it is. So long as it's readable and understandable, it is the content that matters.

The PRD is comprised of four major areas.

Product Purpose

Your job is to paint the *target*. The team needs to know what they are aiming at, and that target needs to be described as clearly as possible. Make sure your description covers:

- The problems you want to solve, not the solution
- Who is the product for? Companies, Customers, Users
- Details are great, but the big picture must be clear
- Describe scenarios

Remember that while brainstorming sessions and oral instructions and discussions lead to better written requirements, they will be lost if they don't get into the PRD.

Features

The body of the product requirements specification will of course be the requirements themselves. The specifics of the requirements will entirely depend on your domain, but regardless of your industry, your product team will benefit from clear, unambiguous requirements that state the need, rather than the solution.

Describe each feature at the level of the interaction design and use cases. You must be very clear what each feature is and what the user experience should be, while leaving as much flexibility to the engineering team as possible.

It is also important to identify which of your requirements are in support of each objective. This is part of the discipline referred to as "requirements traceability" and for mission critical products this is often a formal process. But every product specification can benefit from clearly identifying exactly which requirements support which product objective. If someone decides to cut a requirement, it can be difficult to understand the full impact of this cut. The mapping from requirements to objectives helps make this clear.

Release Criteria

The release criteria are often just hand-waved, but a good PRD puts considerable thought into what the true minimum requirements are, for each of the release criteria, which typically includes:

- Performance
- Scalability
- Reliability
- Usability
- Supportability
- Localizability

Schedule

One of the most difficult sections is to describe the timeframe that the product is needed. It is not useful just to list a random date, but rather you should describe the context and motivation for the timeframe, and describe a target window.

You want the entire team to be as motivated as you are to hit that target window, with the right product for the market.

Step 8: Prioritize

Beyond clear requirements, it is important to prioritize and rank-order every one of your requirements. Most product managers, if they prioritize at all, simply indicate whether a requirement is "must-have," "high-want" or "nice-to-have" (or some comparable classification system).

The classification is important and must not be taken lightly. The product manager should have an extremely high bar for anything flagged as "must-have." This literally means that the product should not ship if even one of the "must-have" features is not ready. So extreme care should be used for any feature marked "must-have" and these features should map directly to the core value proposition of the product.

The "high-want" features are important, and you do want all of these in the product before shipping if at all possible, but you are not willing to delay the product for these features.

The "nice-to-have" features are useful for the product team to be aware of, even if most of them do not get built until a future release, at least the product can be architected to handle these features, where appropriate, in the future.

While the classification of requirements is necessary, this is not sufficient. Within each prioritization classification it is important to rank-order each requirement, from 1 to n. There are a couple reasons for this.

First, time to market is always a concern, and schedules often slip, and you may well be forced to cut some features in order to get to market. You do not want the product team to implement the easy features first, and end up with several not so important features ready to go, yet customer-critical features on the chopping block.

Second, during the design and implementation phase, the team will learn a great deal as issues arise and are resolved, and it is very possible that you will identify additional critically important features to be added. The prioritization helps you to know what to cut in order to accommodate more important features.

The point is that if the product manager does not prioritize and rank-order the features in terms of their importance in creating a successful product, then other less relevant factors will influence what gets built in what order.

The entire PRD process should be a continuous refinement and sharpening of your thinking. Sharp thinking equals sustainable vision. Fuzzy thinking equals failed product. It is much easier to decide now than in the heat of the battle, and it also helps engineering come up with a development plan. Recognize that time may change the priorities, which is fine, just be sure to make the change in the PRD.

Step 9: Test Completeness

Now that you have a draft of the full PRD, you will need to test the PRD for completeness. Can an engineer get enough understanding of the target in order to get the product there? Can the QA team get enough information to design a test plan and begin writing their test cases?

Once the stakeholders have all reviewed the PRD and identified any areas that need additional details or clarification, and you have addressed these issues, you now have a PRD to build a product from.

Step 10: Managing the Product

During the course of implementing the product, there will be countless questions identified, even with the best of PRD's. Resolve all questions about requirements by pointing people to the PRD. If it's not in the PRD, put it in the PRD. Your job is to quickly resolve all the questions and issues, and to record these decisions in the PRD.

If you've done your job and kept the PRD accurate, any project reviews that come up should be very easy to prepare for by just selecting from the appropriate sections of the PRD.

Remember that the PRD is a living document, and you should track all features in the PRD through product launch.

Finally, you may find that you need additional topics covered in the PRD. If you believe something additional is required in order to meet your objectives in the PRD then by all means include it.

COMMON PITFALLS

Creating clear, useful product requirements is much more difficult than it sounds, and there are several traps that the product team can easily fall into. The purpose of this section is to describe some of the more common pitfalls and how to avoid them.

Usability Testing Too Little Too Late

If you have never done usability testing on your products, you are in for some surprises. On the one hand, you will see how badly real customers struggle with your product, and that will shake you up. On the other hand, you will quickly see what you need to do in order to address the issues.

The most important things to remember are:

- Plan on doing multiple rounds of testing.
- Do this testing during the product requirements stage and not when it is too late to make the necessary changes to address the issues.
- Be sure that the product manager and designer are present for most if not all of the testing.
- Pay more attention to what the people actually do versus what they say. This is just human nature. If you're not absolutely sure why a test subject did something, ask why, always trying to get them to think out loud, but also realizing that what's most important is what they do or don't do when interacting with your product.
- It can also be valuable to invite key architects, engineers or managers to observe some of the testing as well, as it can be sobering for them to see what the target customers are really like.
- If you do not have usability engineering specialists on your team, you can either contract them from a firm that specializes in this, or you can have your product designer perform the usability testing. While not ideal, as it is hard for anyone to see people struggle with the result of their labors, a good product designer knows how to run the testing, and the results will still be extremely valuable.

You Are *Not* Your Customer

An all-too-common mistake, especially in high-tech companies, is to assume that if you like your product, then your customers will too. This approach was established with the best of intentions.

Back in the 1960's, Hewlett-Packard espoused the "Next-Bench" strategy. This originated from the fact that back then, HP created engineering equipment and tools for other engineers, who worked at these big engineering benches surrounded by piles of equipment. The theory went that if you could come up with something that was useful to the guy sitting at the bench next to you, then there was a good chance that the engineer that down the road at Lockheed or Westinghouse would also benefit from it.

Given HP's business back then, there was considerable truth to that, and in limited businesses like high-end software development tools, there is still truth. But back then HP wasn't building mass-market products.

Yet today you will still often hear the refrain "we eat our own dog food!" The origin of this dubious phrase is less clear, but it is supposed to convey to the customer that we believe in this product so much that we're even willing to use it ourselves. While this may sound obvious, during the dot com heyday of the late 1990's there were many companies whose products were so bad that they never had any actual customers, even the companies that made them.

But assuming that you are in the business of building real products that large numbers of real people will use, it is critical to realize from the outset that there is a world of difference between those of us that build products and those that buy these products and try to use them every day.

While it is very valuable for you to use your own products to the maximum degree you can, you must never be fooled into thinking that just because your co-workers like the product that it will have any degree of success in the general marketplace.

In fact, sometimes the opposite is true. Later we will discuss TiVo and their competitor Replay TV. Many in Silicon Valley prefered Replay's greater degree of control and expandability, yet the clear winner with the broader market was TiVo with their simplicity and ease of use.

As another example, among the technological elite – which more often than not include the press and industry pundits – there is virtually unanimous disdain for America Online's e-mail system. Yet mass-market consumers had little interest in the feature-laden systems that competitors offered, many times for free, and eventually AOL's competitors followed suit.

Some might argue that while this is true for mass-market consumer products, for enterprise products there are far fewer differences. However, it pays to keep in mind that even in the IT departments of major companies, the employees are rarely as immersed in technology and change as we who create the products are. Often IT professionals have been recruited from other parts of the company, from finance positions or customer service, and many have no formal technology education – just what they've learned on the job.

This all relates back to truly understanding your customer. If the objective is to be able to expand the customer base past the early adopters into the broader market, then from the very beginning, you must constantly do a reality check with your product – is this something that my target market will understand? Is it something they will value? Will they really use it?

What versus How

It is very easy in the product requirements to intentionally or unintentionally dive down into the actual solution rather than limiting the requirement to the problem to be solved. The product manager might do this intentionally if she has strong beliefs about how a problem should be solved, or what technology should be utilized. But it is also very easy to do this unintentionally because it is in our nature (and our customer's natures too) to try to think in terms of solving problems.

However, there are several dangers in including the solution (the "how") in the requirements.

First, it is extremely important to be very clear on the problem being solved versus the solution itself. Many different solutions could be addressing the very same problem. Without stating the requirement in terms of the problem to be solved, the engineers and others will make assumptions that might not be correct.

You can sit down with ten customers and hear them describe ten different things they would like you to do to your product. But, for example, only after working with them to understand the problem they are trying to solve, might you realize that they are all simply trying to get around the fact that an operation takes too long, so the root problem to be solved may be one of performance.

Second, predetermining the solution can end up shutting doors to far better solutions to the same problem, regardless of who comes up with the solution. If you can avoid locking yourself into specific solutions for as long as possible, it allows new technologies to be explored and innovative ways of applying those technologies to be discovered.

Third, having the product manager specify the solution can frustrate and undermine the product development team – the engineers and architects who are actually paid to come up with solutions that they must then implement and maintain. The solution is their contribution to the product, and they are the ones that have to live intimately with the solution, so they often justifiably bristle at requirements that read more like a recipe.

For every requirement in the specification, the product manager should ensure that it is clearly stating the problem to be solved, rather than the solution itself.

Too Little Detail

A different sort of problem is when the requirements are underspecified. It takes a lot of work and thought to fully specify the requirements of a product. But if areas are not covered, then bad things can happen.

The product development team may assume something is not required while it actually is. Or, everyone – the product manager, the engineers, and the testers – may all assume different things.

It is sometimes difficult, however, to realize that you haven't specified something until the product team starts building. You can minimize this problem by having the product development team review drafts of the specification early and try to flag missing or underspecified areas. But, realistically, you always should plan on having to work on some areas during development that nobody anticipated or identified.

Too Much Detail

The reverse problem is also possible. The product manager decides to provide so much detail that the result is a specification so massive that the product team doesn't even read it, or it may take so long to produce that it becomes obsolete before it is even done.

This of course depends a great deal on the type of product you are building. A car or plane will necessarily have one level of rigor required for specifications, while an internet service may have another.

Less Is More

Businesses and business people very naturally think in terms of more is more. If you want to reach a new market or address a new need, you add people and money and you add the necessary product features. While there is some truth to this, it is also true that with high-tech products, adding features can actually lead to less. Adding features adds complexity. It adds customer support costs. It adds maintenance costs. Fewer people can figure out your product. Fewer people see the point of the product.

As Jim Barksdale, former CEO of Netscape Communications used to say, "The main thing is to keep the main thing the main thing." Focus is essential. Adding features makes it harder to keep that focus.

This is not to say that we should never add features. Continuing to evolve and grow a product is essential to continuously meeting and exceeding your customer's expectations. But it does say you must be extremely disciplined about what is added.

Many products have started out good and then have grown so bloated that they no longer solve the original problem as well as they used to.

Engineering-Driven Requirements

There are two extremes of another spectrum that can both cause problems with the requirements and result in unsuccessful products. One extreme is when the requirements are too engineering-driven, and the other is when they are too market or customer-driven.

In terms of engineering-driven requirements, there are definite and real needs that the engineering team has regarding the product. They will be building and maintaining the product, and if the product breaks they'll be the ones up all night trying to fix it. It is also important to realize that good engineering teams are constantly learning new technologies and exploring potential uses of those technologies.

That said, there is a danger among some teams of the engineering organization overwhelming the product manager with requirements and essentially defining the product they are to build, either based on a technology they are excited about, or because they are challenged by a particularly hard problem they want to solve.

This is where the focus on value can be so important. The product manager must listen to the needs, but she is ultimately responsible for defining a product that customers will buy.

The point is to define good products that can be executed with a strong effort – versus good products that can't be executed or let engineering build whatever they want (e.g. solve the hardest problem).

Customer-Driven Requirements

The other extreme is where the sales or marketing organization dictates the product requirements, usually as a result of discussions and negotiations with customers.

Bill Hewlett, co-founder of Hewlett-Packard, used to emphasize "as important as they are, marketing people must play a secondary role in the question of product definition." He understood how hard it is for customers and sales and marketing people to specify a successful product.

The product manager needs to understand deeply what problems the customers are trying to solve, and have room to be able to come up with a product that will meet the needs of a wide range of customers.

One especially dangerous form of customer-driven requirements is called a "special." This is when someone at your company – the product manager, an executive, a sales manager – promises one or more customers a special version of the product, or the addition of specific features, in exchange for purchase or for additional fees.

Especially with young companies, this is a very common strategy. The argument is typically made that they are simply responding to customer needs, and the customer is simply underwriting product development that would likely have been needed to be done anyway.

While this sounds relatively harmless, for a product company, this is a very slippery slope, and one that you need to do your very best to stay off of.

There are several issues with specials. First, the nature of the product business is that you are trying to come up with a single product that can be used successfully by many customers. Anything that works against that goal is dangerous.

Just because a customer thinks she wants a specific feature, it's not necessarily the case that she will want it once she actually gets it, as customers often struggle when they try and come up with the solution to their needs, rather than just try to identify the origin of the need itself so that the product team can look at the best way to meet that need. Further, the requested feature may not be useful for other customers, and in fact could end up burdening and confusing the product for everyone else.

If the customer wants her own version of the software, that is also problematic; while it doesn't directly hurt other customers, indirectly there is an impact as the organization has to now build, test, and maintain multiple versions. It does not take many specials before the organization is so bogged down in specials that it can't effectively pursue its main business.

There is a whole industry that exists for creating custom solutions – these companies are either specialty houses, or systems integrators. While many have tried, it is no coincidence that few of these companies have succeeded in creating successful general products.

To avoid this pitfall, your company needs to be disciplined and in fact willing to walk away in certain cases, and you also need to ensure that your standard product is useful and compelling enough that your customers can live without special custom features.

If you simply must support custom features, consider enabling your product to be tailored by the system integrator companies and refer your customers to them for the customization.

Emotion Can Be Logical

Another common mistake is to underestimate the role that emotion plays in a successful product. And this applies to not just consumer products; corporations also make purchase decisions based partly on emotion as well.

Often product managers come up with lists of desired features, and the engineering team estimates how long it will take to build. The product manager then prioritizes the features and if it's not considered essential, she cuts them out or defers them to follow-on releases. While this is reasonable, it all boils down to what is essential. Often the initial product that comes out is dull and uninspiring, and it is questionable if the team will even get the chance to build out the rest of the product, which could have made a much different impression.

This is not an argument to include all the features that the product manager can think of. In fact, most products have too many features. Rather, the point is that the strong product manager understands which features will provide that essential emotional response, and understands that those are among the critical features.

Identifying which features actually will provide that "secret sauce" can be difficult. It comes from having a deep understanding of your customer, and an insight into what will inspire, and elicit the emotional response. Again, prototype product concept testing where you watch people actually interact with your product can provide the key here.

If nothing in your product causes that response, then you may need to try out some new ideas on target customers. What is it that empowers them? People want to be a part of something special – can you tap into that?

Standard Is Better Than Better

One major source of frustration for customers and a source of unnecessary resource investment is when products have gratuitous differences for standard features and operations. When you encounter this you know it. It's the rental car that has the door locks in a different place and you can't find them at night, or the cell phone that doesn't dial the same way all the rest do. Does the car work better, or does the phone dial better? Not really. Gratuitous differences just detract from what innovations are actually present.

Be very clear that this is not about innovation – but good products innovate in areas that people have real need. They do not innovate in areas of existing standards just for the sake of providing something different. For most customers, it really is true that "standard is actually better than better."

If there is any way in your product definition that you can leverage what your customers already know and be consistent with that, they will appreciate your product all the more for it, and leave them much more open to trying your new innovations out where they are truly important.

The point here is just to beware of gratuitous differences, and focus your innovations where they truly add value.

Pay Your Taxes

One of the common pitfalls, especially with follow-on releases of products, is that after several releases of continuously growing your product and adding new features, the engineering team wants to re-architect the product because of issues with maintainability, scalability, security, performance – any number of valid issues. They argue, typically very persuasively, that they can't build any more new features because it is all "on a house of cards," or "it was never designed to support what we're now doing" and they must now take the time to get their house in order.

Assuming they are not simply exaggerating, then the product is very likely in serious trouble. Few products survive such a re-architecture, mainly because customers don't see the value in these changes – they just think that nothing is happening and their needs are not being addressed for extended periods of time. And all too often these re-architecture efforts turn out to take far longer than anticipated. You can tell customers what you're doing, but they don't care – this dirty laundry is an internal problem – not something they want to care about.

The key is to avoid getting into this situation in the first place.

Some percentage (think along the lines of 10-20% of resources, but this can vary depending on the circumstances) in every release needs to go to the engineering team for them to continuously perform the systems work required to keep the foundation solid. If any of their changes will result in something customer-visible, then you as product manager should be involved, otherwise consider this like paying taxes. Just pull the percentage right off the top for each release, and tell the engineering team that these resources are theirs to use each release for addressing their architectural issues.

Requirements versus Design

One difficult issue in requirements specification is whether to include the design in the specification or not. The requirements clearly speak to the "what" and the implementation clearly speaks to the "how," but right in the middle is the design, which describes what the product should look like and the desired interaction with the user.

Note that in this context "design" is referring to the user interaction. It is not referring to internal system or program design.

The argument for including design in the requirements goes like this: The requirements and the design are inherently intertwined, and together make up the user experience. The design will impact the requirements nearly as much as the requirements impact the design. If you don't include the design, then the requirements will need major revisions as soon as the design is done. Further, you will learn a great deal about what the product should really be from the design process. Finally, a requirements specification that includes the design is much more useful to the product development team as they have a much clearer understanding of what they need to build.

The argument against including design in the requirements goes like this: Once the requirements are complete, there are several activities that can get started, including the design, the architectural work, implementation work, test planning, and so on. If you delay requirements completion until design is complete, then you push out the whole product timeline. Further, design isn't the only thing that may impact requirements; the implementation issues may do that as well. Better to get us all started and identify all the issues sooner rather than later.

Both arguments have a great deal of truth to them. The overriding factor, however, for most cases is that once architectural or implementation work begins, it is very hard to back up and make changes. There are many reasons for this, some technical, some cultural.

Unfortunately, if the design process uncovers issues that require changes to the requirements, which it surely will, then it is essential that the requirements be able to change, even if this causes major perturbations to the engineering team. This is why it is preferable, if at all possible, to be able to hold off on architectural and implementation work until after the design is complete.

The higher the cost of actually building the product then the more desirable it is to finish the design first, as this will ensure better products sooner in the product lifecycle.

My experiences at eBay converted me to the camp that includes design in with requirements, and holds off beginning implementation until design is done. With the power of today's prototyping tools, it is very fast and inexpensive to create very useful product simulations and get extremely valuable user feedback while it is very easy and cost-effective to do so.

SUMMARY

One final point on creating good product requirements: During the process of defining the product, it is important to always focus on delivering superior value to the market place. It is easy to get distracted by competitors, vocal customers, and architectural issues, and you do need to understand those needs, but in defining a good product, always remember to focus on the value.

SVPG Silicon Valley Product Group

About the Author

Martin Cagan is the Managing Partner of the Silicon Valley Product Group. During the course of the past 20 years, Martin Cagan has served as an executive responsible for defining and building products for some of the most successful companies in the world, including Hewlett-Packard, Netscape Communications, America Online, and most recently as VP Product Management and Design for eBay. The Silicon Valley Product Group (www.svproduct.com) is dedicated to serving the needs of the high-tech product management community by providing content, services, and professional development for product management organizations worldwide.

1340 S. De Anza Blvd. #102
San Jose, CA 95129
408•410•7334 *(T)*
408•725•3909 *(F)*
info@svproduct.com
www.svproduct.com

SVPG

Silicon Valley Product Group