Product Arts

4 Common Requirements Issues

Part 2 of the Product Requirements in a Nutshell Series

The main purpose of the Requirements process is to communicate to the technology team about a problem solve and the envisioned features and functionality required to provide a solution. While on the surface this seems simple enough, the reality is there are several wrong steps to make along the way that can significantly reduce the value proposition that gets delivered to the marketplace in the form of the product.

This article is the second in the "Product Requirements in a Nutshell" Series and discusses four common issues that can arise in the development of Requirements and the potential outcomes that can result. For more background on typical Requirements documents, see Part 1 – A Tour of Requirements Documents.

As discussed in our first article, at a high level the requirements process is attempting to answer: 1) What are the Market & Business Drivers for the product release; 2) What are the User's goals and how will they use the product; and 3) What solution are you going to build?

The four issues discussed in this article are:

- Missing Market & Business Drivers
- Focus on Features, not on User Goals
- Missing Qualities and Constraints
- Lack of Market Validation

#1 - Missing Market & Business Drivers

The MRD (or PRD or BRD) needs to set the context for why you're doing the product or project at all. It aligns two fundamental pieces of information for the development team:

1) What is the real problem you're trying to solve in the market and the envisioned needed feature set to solve it, and
2) In solving it, what is the desired outcome for the company

Without an accurate view of the problem and an envisioned solution that creates value for customers, the success of the product is probably curtailed from day one of the project. This is arguably the #1 reason for product failures across the board. The solution is either looking for a real problem to solve or the problem is only partially solved yielding reduced or possibly no value to the customer.

One example is Segway. The nifty technology came out of space program research and resulted in an amazing and entertaining product. While it has seen some limited success in vertical markets applications such as security personnel carriers and warehouse use, as a consumer product it has failed. The reason? Does it really solve a need better than existing alternatives? Current marketing as a golf cart replacement begs the same question – what's wrong with current solutions that this improves on.

The alignment to current company objectives with measurable outcomes answers "so what?" and provides reinforcement that it matters. The objectives can be lofty, such as "Become the leading provider of X within 12 months", or less ambitious, "Increase our market share in X by 5 points in

12 months". Even for product updates to an existing product, rallying around a major theme that addresses a sizable problem for users and is expected to improve or support the current trajectory of results will be a more powerful initiative. For example "Decrease customer care calls by 50% due to installation errors". It is not uncommon to see software releases that are composed of a long list of unrelated and often small improvements and as a whole is delivering little value to the marketplace or to the company.

In addition to identifying the real goals you're trying to achieve, alignment of stakeholders around priorities for the release is also required. Assuming the major stakeholders agree on the market problem and business objectives to be achieved, prioritization of the features and functionality become immensely easier. Most importantly, it helps to instantly identify what you are NOT going to do. Any feature that does not support the release objectives goes on the back burner. No negotiations, no arguments, end of discussion. If there isn't stakeholder agreement, then of course there will probably be challenges all along the way.

2 - Focusing on Features, not on Users Accomplishing Goals

Another way to say this is too much focus on the solution, not the problem. As technologists, we're in love with technology and solving the problem, so conversations quickly jump to the cool implementation. It's easy to lose the user problem view and thus easy to fool ourselves into thinking we're actually providing value by our solution. This is probably easier to show in an example than to discuss.

Suppose we're developing the first generation of ATM machines. Our product requirements could easily focus on "features" of our product: LCD display, keypad, card reader, receipt printer, cash dispenser, and deposit slot. From a user's viewpoint, these are meaningless pieces of technology and the final assembly and implementation may end up being a terrible experience in accomplishing what they want to do with it, which is to do their banking. The table below compares a Feature level focus versus a Goal level focus.

Feature Level Focus	Goal Level Focus
✓ Simple display and keypad operation ✓ Secure bank card reader ✓ Cash dispenser ✓ Deposit accepter ✓ Receipt printer	✓ Secure and confidential access to banking services at a time and location convenient to you ✓ Perform the activities you need: get cash, make deposits, transfer funds, check your balances ✓ Protect your transactions through secure access and positive identity confirmation ✓ Keep a record of all your transactions

ATM Machine – Features vs. Goals

The product requirements are best served by focusing on the Goals as the highest level and the Features as the response at a lower level and traceable to the Goals.

Another reason for focusing the requirements at the Goal level is to identify the desired user experience and to spur innovation during the development cycle. If developers understand the true goals and motivations of the users, it enables them to think creatively about how to make the experience better, versus just complying with delivering a set of features in a requirements doc. This is the difference between building an MP3 player and building an iPod. At a feature level, it may be impossible to distinguish the requirements between these two products. It's at the Goal

level of what the user wants to accomplish that separates winning products from technology solutions.

#3 - Missing or Poorly-defined Qualities and Constraints

The Qualities and Constraints are also called the Non-Functional Requirements of a product. It is usually the omission of these requirements that cause issues, and assuming that the developers will just do the right thing. Unfortunately, the right thing could be many possibilities to different people. The other issue is testability of the requirement. The requirements need to be measurable in some way for testing.

The Qualities of a product define how well it needs to perform. From a user's perspective, these primarily revolve around usability and performance, but also include basic expectations that the product just behaves "as it should", such as reliability and availability. One issue that crops up is a complex and confusing user interface. A common requirement would be to state "The UI must be simple and easy to use" however this is not testable because it's subjective and arbitrary. An alternative could be stated as "A Typical User X can perform operation Y within 2 minutes, as demonstrated by usability testing". This forces a proactive approach to the design and provides a measurable test that it passes or fails.

Performance and usability have a strong linkage, especially in server-based software systems. If the system bogs down, or worse – crashes, due to too many simultaneous users, this affects the usability of the system. Specifying realistic forecasts for user adoption coupled with usage profiles of the operations they will be performing will help the designers scale the system appropriately.

Constraints are restrictions placed on the system and limiting it in some way. Some constraints simplify the designers' and testers' jobs by limiting the breadth of support required, such as for certain operating systems, browsers, or existing system compatibility. Other constraints make it more difficult for the designers, such as size or memory footprint required. Some of the major constraints may be part of the high level feature set, while others may go unaddressed and are not discovered until very late in the game. These can include security requirements, data privacy, certifications, operating environment, external system compatibility, etc. In this case, the lack of proper market requirements in a specific area can be a potential deal-breaker for some significant sales opportunities after release.

For both Qualities and Constraints, the primary issue is to understand fundamental assumptions and expectations your target market has about your solution and deliver it to them. You can't always identify these expectations by simply asking your customers. You need to have a deep understanding of their business, processes and goals to be able to fully anticipate their expectations.

#4 - Lack of Market Validation

Our last major issue to discuss is one of the most powerful and easiest practices to employ to predict the potential success of the product or release, and yet is not all that common. It involves validating your completed requirements with some market representatives. This can be accomplished before a single line of code or engineering drawing is developed. If you're doing a product update, it could be as simple as a visit or teleconference with some lead customers to review the document. In some scenarios, you may be trying to reach a new market, and thus will

need to find some non-customers to solicit their input. While not very exciting to review a document, even this will provide feedback to you.

For brand new products, a prototype is going to be significantly more successful at flushing out market input. This can be a simple mockup or paper presentation, or a limited software implementation in Flash, or it may be a full working demo. The higher the risk and investment of the product, the more you will gain from taking the **short relative time and cost** required to implement the validation. The mockup or demo will also get miles of use long after development is started as you continue to communicate the product internally and externally to partners and potential customers.

There are a dozen reasons why companies choose not to validate their requirements. Here are a few:
- "We don't have time" (*...to spend 2-4 weeks of the next 12-18 months plan to get the product right*)
- "We don't have the budget" *(...to spend $20k-$50k out of the $1M-$2M ask to get the product right)*
- "We're in stealth mode" *(...and it's more important to surprise the market than to get the product right)*
- "We already know we have a hit" *(...because of our track record of predicting successful products)*

Maybe the reasons are valid. Just ask yourselves --- "What would you do if you were personally funding this product from your own wallets"? Is it worth spending a couple years of your life to get it wrong?

Summary

This article presented four commons issues that can derail the requirements process and potentially the entire product. This list was not intended to be all-inclusive, and there are several other areas to misstep in the requirements process, with a few being Poor User Experience, Wrong Level of Requirements Detail, and Focus on HOW Instead of WHAT. However, the issues that have been discussed focus on getting the primary value proposition of the product release figured out before diving into second level issues of the requirements process itself.

About Product Arts

Product Arts specializes in Product Management consulting and training. Our training includes public and private custom training on Product Requirements. For more information, go to www.product-arts.com or email info@product-arts.com.

