The Leading Reference for technology-based products

Product Management Journal

Volume 10 £10/€12/$14
Updated

Contents

Requirements

How to make sure they deliver

product focus

Training

Product Management and Product Marketing

Industry-leading public and private courses

Live online and face-to-face delivery

Focus on technology-based products

Tools, templates and checklists

Learn best practice

Certification

product focus

www.productfocus.com

Welcome

Leading the way for product managers

Requirements is a big topic. In recognition of this fact, we've produced this updated version of our original Requirements Journal with more articles, more insights, and more best practice.

Writing requirements is a large part of many product management jobs. We spend a lot of time trying to understand what they should be, writing them down, reviewing them, prioritizing them, and checking if they've been done.

The end goal is delivering what customers will value as quickly as possible and in a way that makes money for our business.

We hope you find this journal useful, and if you want to find out more, why not attend one of our training courses?

Good luck with your requirements!

Who's who?

The *Product Management Journal* is published by Product Focus as an independent publication for product managers with technology-based products. Product Focus was founded and is run by Ian Lunn (top) and Andrew Dickenson.

The founders continue to deliver many of Product Focus' training courses and reviews alongside their team of senior consultants.

To get all our previous journals, and receive the latest copy, sign-up at **www.productfocus.com**

Product Management Journal
Launching
Hit the ground running and what not to do
Propositions
How to build sharper propositions that really work
Business Cases
How to create outstanding business cases that deliver results
Pricing
Setting the optimum price Tips, tactics and theory
Leading
Product management and product marketing teams
Agile
And its impact on product managers
Market Analysis
Understanding markets, customers and the competition
Roadmaps
Compelling roadmaps that align everyone to your vision
Requirements
How to make sure they deliver
Training
...why it's needed and how to make sure it delivers
Strategy
Setting the foundation for effective product plans
Take control
Putting product management in the driving seat
Product Marketing
Getting products in front of customers and making sure they sell
In-Life
Making sure your product delivers when it matters

All the trademarks and tradenames referenced in the Journal are the property of their respective companies

Requirements
The big picture

There are 3 big things to get right if you're working on requirements.

The first is to involve customers throughout the development process. Talk to them to understand their problems. Check you're on track when they review prototypes and test early releases. Involve them in trials before you launch. These are all essential if you want to build a product they want.

Secondly, product development works best when there is close collaboration between the key disciplines involved, e.g., Product Management, Development, Operations, Manufacturing, and Design/ User Experience experts. This involves an ongoing conversation about what's needed, possible, and feasible.

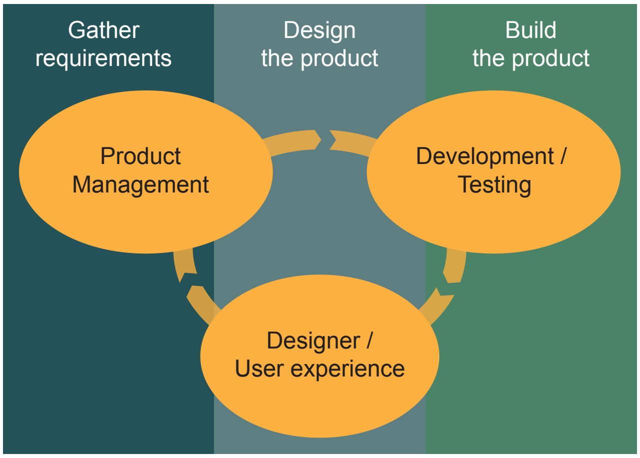


Fig. 1 The development steps, key roles and interfaces

Finally, there are various ways of writing requirements, and it's important to pick the one that fits with your development approach and the type of requirement you have – to make life easier for everyone.

Fundamentals

Whether your company uses a Waterfall methodology such as Stage-Gate, an Agile approach like Scrum, or some combination, the basic activities remain the same. These are to gather the requirements, design a solution, and build a product.

However, as a product manager, your first job is to understand the market problems and decide which ones make strategic and financial sense for your business to solve. If the business buys into your vision, then budget and resources are made available. Once you have the commitment, you can gather, prioritize and communicate the requirements to the people who will design and build the product.

“In several projects I have run, there seemed to be more focus on hitting the target release date – rather than the product manager checking, we were actually delivering a viable product.”
Jeroen Visser, Consultant

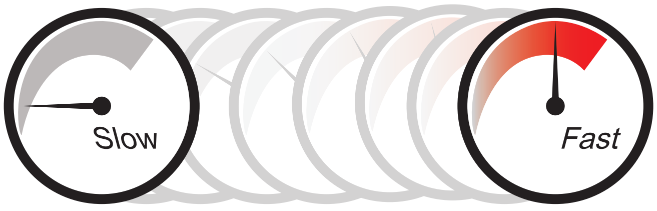
An ongoing process

Managing requirements is not a one-off task. In reality, there is an ongoing stream of requirements that has to be managed throughout the development and beyond. As you uncover and adjust requirements, you’ll be continually balancing what to include in each release, given the resources available and your target delivery date. For example, what goes on the roadmap for the future? What just isn’t important enough for your target market? What really does have to make it into the next release?

There is no single best way

There is no right or wrong way to write requirements. The more formal and detailed the process, the less room there is for ambiguity, but the longer things take. Too little rigor, and there is a danger that what’s built doesn’t meet market or business needs. What’s appropriate will depend on your product, company culture, what’s been successful in the past, development processes, proximity to the development team, time-to-market pressures as well as your appetite for risk.

You need to decide where on the scale you want to be (see Fig. 2). For example, you would be well to the left for a life-support system and well to the right if looking after a small website for your local club.



Heavy process	No process
Stringent standards	No standards
Heavy documentation	No documentation
Rigorous reviews	No reviews
Methodical & detailed	High level
Strict sign-offs	No sign-offs
High traceability	No traceability
Low risk	High risk

Fig. 2. Based on diagram from ‘Just Enough Requirements Management’ by Alan M. Davis

So what could possibly go wrong?

As we write requirements, we get things wrong and miss things out. Things get miscommunicated or misunderstood. Our understanding of customers and their problems improves. And the inevitable pressure to

"Product managers can be a bit removed from the customer with the salesman in between, so information gets distilled or distorted."
Sonja Jeffery, Product Marketing Manager, Fujitsu

release quickly means that things are dropped or changed. So what gets built is rarely exactly what we initially expect or finally want.

The famous tree swing cartoon below shows the dangers of failing to understand, communicate, and check customer requirements properly. The less discussion and more hand-offs there are, the more these problems are amplified.

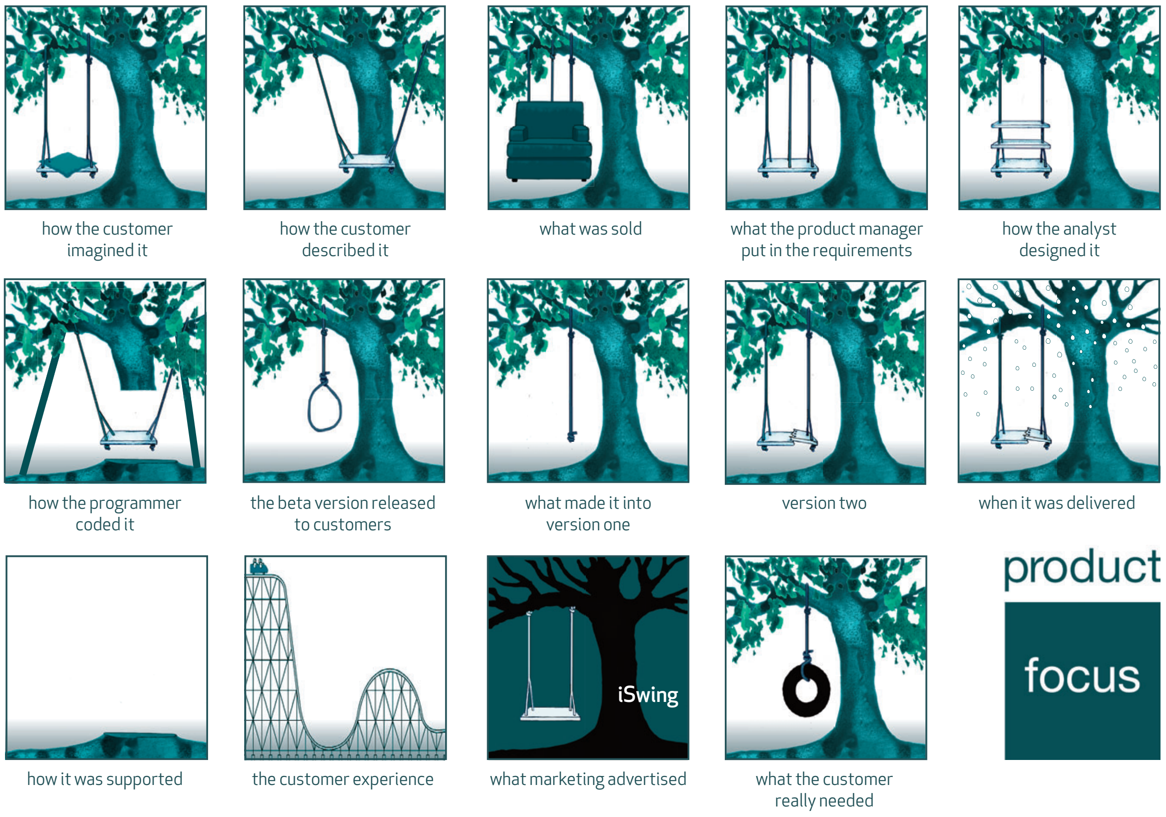

how the customer imagined it
how the customer described it
what was sold
what the product manager put in the requirements
how the analyst designed it
how the programmer coded it
the beta version released to customers
what made it into version one
version two
when it was delivered
how it was supported
the customer experience
iSwing
what marketing advertised
what the customer really needed
product focus

Fig. 3 Swing diagram – original source unknown

Summary

As a product manager, you need to ensure the requirements process builds a product that people will buy. Involving customers from the start, cross-departmental collaboration, and writing requirements in a way that works for everyone are all great ways of making sure this happens.

Who does what?

Product Manager vs Product Owner vs Development vs UX

There's often confusion between the roles of Product Manager and Product Owner. Who does what? Who writes the requirements? And what about UX (User eXperience) or Design and Development who also get involved in writing some types of requirements?

Why is there confusion?

If a business is using Scrum (the most common Agile approach for software development), one of the defined roles that need to be in place is a Product Owner. If there is also a product management function, then often the big question is … well, how different is the Product Owner role to that of being a Product Manager?

"Your most unhappy customers are your greatest source of learning."
Bill Gates

From our 2022 Product Management Survey, we know that a third of Product Managers who work in companies using Scrum are also Product Owners.

We also know from the people attending our training course that many Product Owners do product management activities. In fact, in some companies, Product Owner is used as a job title when they don't even use Agile. No wonder it's confusing!

What does a Product Manager do?

A Product Manager leads the planning, delivery, and marketing of a product over its full life-cycle. They do this by working with many teams across the whole company. To be effective, they need a balanced view across all the different aspects of the product, e.g., technical, commercial, and operational. They feel ownership of their product and responsible for its success.

What does a Product Owner do?

A Product Owner provides the requirements for the development

Wise Words
Product Managers build the right thing - Product Owners build it right.

team in Scrum. They work as part of this development team to release a series of software deliveries in sprints. Their focus is typically on writing requirements and prioritizing the requirements backlog, and they also have a responsibility to optimize the value of the work being done. They need to be available to answer any questions the development team might have. They also feel ownership of their product and are responsible for keeping the development team gainfully employed.

A tool to help

Our Product Activities Framework can help you understand the difference between the two roles. It shows all the product management activities that should be taking place in a business with products. Not all of these will be done by the Product Manager, but usually, a large proportion are. The Product Owner role (as defined by Scrum) is typically only concerned with the top two Inbound Activities in the green box – Discovery & design and Requirements.

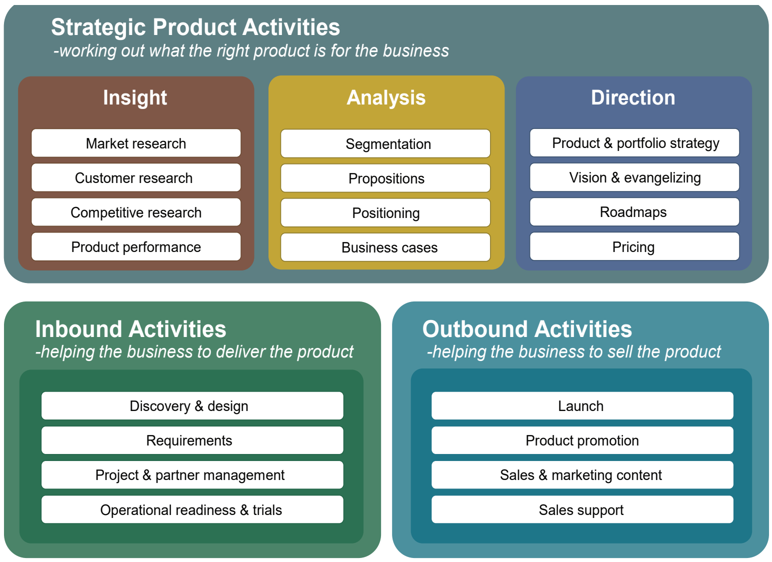


Fig. 4 The Product Activities Framework

How are things set-up?

We come across lots of companies as we do our training and reviews, and what we see are five variations.

The first is the company that doesn't have any Product Owners – perhaps because they aren't developing software or using Scrum. There are just Product Managers.

In the second, the company does have Product Managers, and they are expected to take on the Product Owner role as part of their job.

In the third option, the Product Management role is separated from the Product Owner role. In this case, the Product Owner role exists within the Development or Product organization and is done by, for example, a Business Analyst, Lead Developer, or Requirements Engineer who takes high-level direction from the Product Manager. The Product Manager has the high-level view of the market and customer needs. They own the roadmap and provide the high-level direction to the Product Owner. This is shown as a dotted line on the diagram.

In the fourth option, there is the Product Owner role, but their role is extended from the pure Agile version of the Product Owner to take on some (or all) of the Product Manager's responsibilities.

In the fifth and final option, there is only the Product Owner role with a focus on requirements and working with Development to ensure these are clearly communicated, estimated, and delivered. Any product management activities are spread across the rest of the business.

So how do these options compare when it comes to writing requirements?

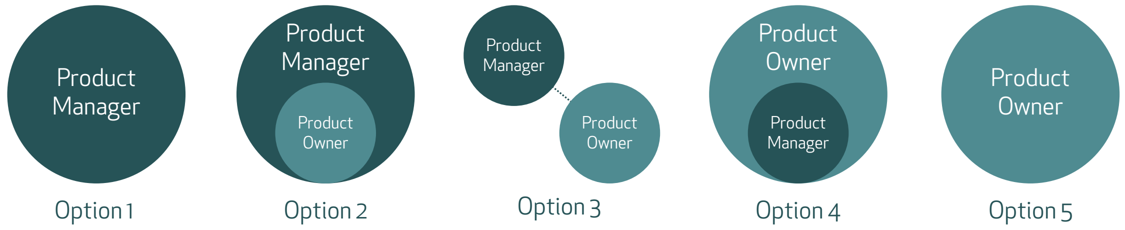


Fig. 5 Product Management and Product Owner roles

Product Manager only (options 1 and 2)

Where there is only a Product Manager, then they are normally responsible for writing requirements.

In Waterfall (the step-wise approach to development) the Product

"One-size-fits-all roles won't work for me and my teams. As a Product Manager, the Product Owners working on the more mature products in the portfolio need my focus on future vision and roadmaps, while the very new need me to get stuck in and get my hands dirty with artifacts like personas and positioning. What a Product Manager vs. a Product Owner does and how they interact will come down to what works best for your product, team, and of course your market."
Joanne Butler, Product Manager, Charles Taylor InsureTech

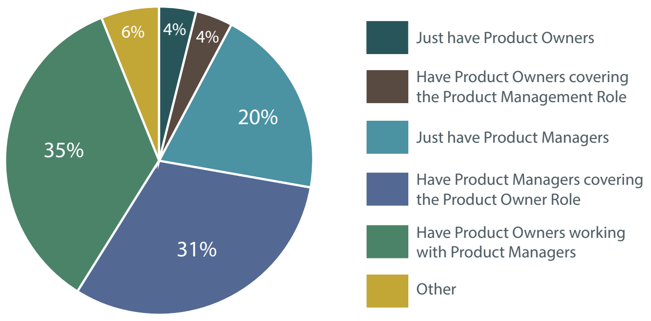


Fig. 6 How do companies set things up? (2022 Industry Survey)

Manager tends to focus a lot of their time on a single stage in the development cycle, e.g., doing analysis or writing requirements. These requirements are written in a lot of detail. Once completed and passed to Development, the Product Manager can move onto other things.

If a company adopts Scrum, the Product Manager may also take on the Product Owner role. This can mean a sudden and dramatic increase in their workload. That's because Scrum demands the Product Owner should be available to talk to Development about requirements throughout the whole development cycle.

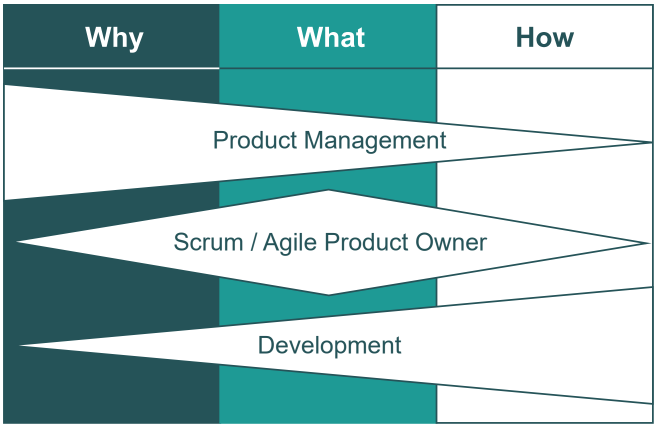


Fig. 7 Adapted from Agile roles framework created by Paul Inness, using the Golden Circle by Simon Sinek

We know from talking to delegates on our training courses that being a Product Owner to one Scrum team takes around two days per week. The Product Manager can become so busy that other parts of their role suffer. And that leads to frustration and worry as they don't have time to do good quality work. In this case, we recommend option 3.

Product Manager and Product Owner (option 3)

In option 3 , someone, often from Development, works as the Product Owner taking direction from the Product Manager. The Product Manager writes high-level requirement; the Product Owner breaks these down into detailed requirements and manages the prioritization of these with Development. The Product Owner and Product Manager must work closely together to align their priorities.

Product Owner only (options 4 and 5)

Where there is only the Product Owner, their role often expands beyond what was originally envisaged.

It's like the 'scope creep' issue for requirements where you're asked, 'Can you just add this… and this?' until it becomes impossible ever to deliver. As the product expert, they might be asked, for example, to help review marketing material, do sales support, and get involved in product roadmapping and planning. The Product Owner takes on more and more and gets busier and busier.

It's easy to end up with a situation where an individual gets overloaded and can no longer do the job expected of them.

Product Manager/Product Owner vs. Development

Another question that may come up is how these roles fit with the Development team.

A simple definition is that the Product Manager/Product Owner knows **what** needs to be built (the requirements) and **why** (the context and justification), but Development are the experts on **how** best to build the product.

The Development team write the architecture guides and requirements and, in Waterfall, the high-level and low-level design documents. In a manufacturing business, other teams, such as manufacturing and process engineers, will also be involved in designing and documenting solutions to deliver what's been requested.

Product Manager/Product Owner vs. UX/UI Designers

The goal of User eXperience (UX) teams is to give users an excellent experience when using the product. What this means depends on your product – for some, the focus is on look and feel, for others, the focus is on enabling the user to easily and efficiently do what they need to do. In the world of software, you sometimes have two roles – a User Experience (UX) Designer and a User Interface (UI) Designer (another potential source of confusion on who 'calls the shots').

The UX Designer's job is to find out what customers need through research and then to use this insight to create and test early designs (wireframes) with representative users. These can then be developed into interactive prototypes that can be further tested and iterated upon. It's typically at this prototyping stage that a UI Designer may get involved. They improve the experience by working on visual design, color, typography, layout, and branding.

So, a simple definition is that the Product Manager/Product Owner knows **what** needs to be built and **why,** but UX/UI are the experts on **how** to create the best user experience.

Summary

Whatever setup you have in your company, people must have enough time to do their jobs properly and are clear on the scope and responsibilities of their role.

If the Product Manager, Product Owner, and User eXperience roles exist, we think it's important you have one person (the Product Manager) making the key product decisions. Based on a balanced view across the different aspects of the product and then giving direction to the rest of the business.

Vision

Inspire and guide

Before you even start on the first requirement, it's important to provide a vision of what you're trying to achieve. It inspires your colleagues, creates engagement, and helps keep everyone aligned.

Who's best placed to write this down?

It's the person with the most balanced view of the product and market opportunity and the one who'll likely be held responsible for product success. That often means that it's the Product Manager.

"Regardless of your industry, it's essential you have a clear vision and understanding about what value your product will bring to its target customers, what problems it solves and why your company is investing resources in creating it. It's also extremely important that the vision can inspire and drive your team and organization into making the product a reality."
Hugo Santa Maria, Senior Product Manager, ADB

What's in the vision?

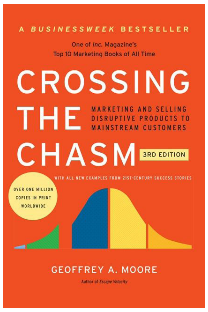
The intent of the vision is to get stakeholders engaged and excited about delivering a clear set of goals for the product. It provides a picture of what could be. This helps keep people aligned with delivering for the longer term while they're dealing with the short-term pressures of building the product day-by-day.

When there's pressure to build something quickly or with limited resources, products can suffer a 'death by a thousand cuts.' Lots of small compromises and shortcuts, none of which are fatal in themselves, together can add up to a slow and painful demise. Keeping the vision in mind helps avoid this.

Challenges

The vision helps frame the future and is a tool to help people make decisions on priorities. To do so, it needs to avoid the biggest pitfall we often see, which is where vision statements are packed with generic feel-good aspirations such as: "The product will be the highest quality… differentiated in its marketplace… deliver focused growth… cement leadership in our target market… etc."

While you can't argue against these being good things, when the vision effectively says we want to be the best at everything without compromise, then it is useless at helping people choose between options. In the real world (at least my world!), being the best at everything doesn't happen. The vision must focus on a limited number of primary goals to give readers the clarity they need to make decisions. It should spell out how you want to stand out in the market place.


A BUSINESSWEEK BESTSELLER
One of Inc. Magazine's Top 10 Marketing Books of All Time
CROSSING THE CHASM
MARKETING AND SELLING DISRUPTIVE PRODUCTS TO MAINSTREAM CUSTOMERS
3RD EDITION
GEOFFREY A. MOORE

Crossing the Chasm
by Geoffrey Moore

What's a good format for a vision?

There's no single format that works for every vision, but good ones commonly include a description of target customers, what they'll value, and how the product will satisfy those customers like a proposition.

A good starting place is the structure created by Geoffrey Moore in his book 'Crossing the Chasm'. It's like the structure of a requirement but is written at a higher level of abstraction and is extended to say how what we'll deliver is different from our competition.

- For (target customer)
- Who (statement of the need or opportunity)
- The (product name) is a (product category)
- That (key benefit, compelling reason to buy)
- Unlike (primary competitive alternative)
- Our product (statement of primary differentiation)

Sales pitch

This also makes a good structure for an internal elevator pitch (a 30-second to 2-minute pitch you might make to someone to explain what the product's about and why it'll be successful). I always found that I needed to sell to the different internal stakeholders when I went around the business to explain about a new product and get their support. The vision becomes your sales pitch.

For Finance Controllers in small and medium-sized businesses
Who want to take away the headache of chasing unpaid invoices
The Finance 2.1 product is a software accounting tool
That makes sure you never miss anything
Unlike other products with manual bank reconciliation
Our product provides real-time integration to all major banks

Don't forget the emotional needs

Depending on your product, it can make sense to extend this vision with extra descriptors for the specific qualities of the product or the emotional response you're aiming to elicit in users. For example, saying that the product should be 'trusted,' 'fun,' 'generate pride,' 'make the user feel in control,' can provide additional clarity for stakeholders.

What's the process to create the vision?

As a busy product manager, it can be tempting to close the door, sit in the quiet, and write out a vision. That's what most of us do, but we think there is a better way. To be effective other people need to be engaged. Doing so means that the vision integrates their good ideas and helps them buy-in to what's been defined.

"Most importantly, I've discovered that getting 'face-time' with the development team and bringing them on the journey really helps to lubricate the requirements process – people are more willing to go the extra mile with you if they understand why you're asking for it."
Michelle Anderson, Product Manager, ADESA UK

Who to involve?

And who should be involved in these discussions? It's always worth remembering the ***'3 whos'*** of meetings. These are ***'Who knows,' 'who cares,'*** and **'who can do something about what's decided.'** The list of participants will vary by company, but typical stakeholders include Product Managers, Product Owners, Product Marketers, visionary technical, sales, and commercial people, as well as those working on User eXperience. Sometimes you might find that your support organizations have some great insights as well; they talk to customers all the time!

Summary

Whatever approach you take to create the vision, you'll need to consider how far into the future you want to go. In small, agile companies, it may only be a year. In businesses developing big products (imagine if your product was a commercial airplane), it might be many decades. It needs to be concrete and understandable, so people can align their plans and make decisions. And it needs to be aspirational with the bar set high enough to really engage and inspire people.

MRD
The Market Requirements Document

Whatever development approach your company uses, as a Product Manager, you need to set the scene and explain what you're trying to achieve.

A key document that is often used to do this is the Market Requirements Document (MRD).

"The main thing is to keep the main thing the main thing."
Jim Barksdale, former CEO, Netscape Communications

Where does the MRD fit?

The figure below shows the logical progression of documentation needed to justify and guide a development project. It starts with the document used to secure the product investment, e.g., the business case. The MRD comes next and provides more detail for developers, setting the scene and providing context. The final stage is documenting the individual requirements in detail. In Waterfall, these are sometimes put into a document called the Product Requirements Document (PRD).

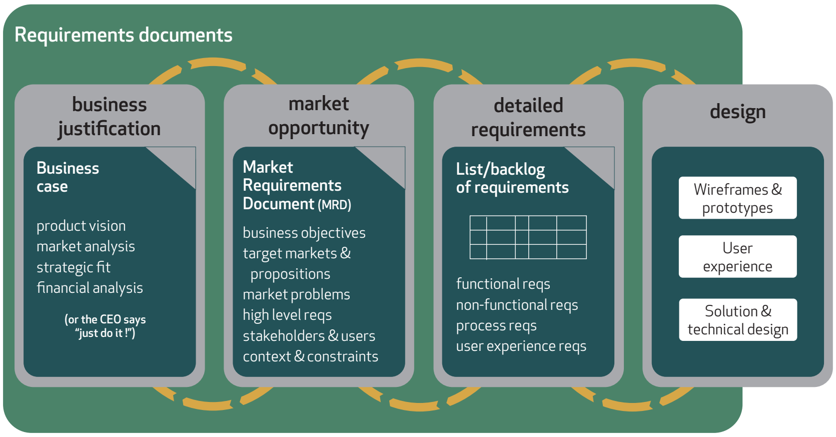


Fig. 8 Typical Product Manager responsibilities

As a Product Manager, you're likely to lead and be writing the documentation for the first 2 stages. For the 3rd stage, you may be involved, or it may be Product Owners, Business Analysts, or Requirement Engineers. In some companies, you may also get involved in reviewing the design in the 4th stage. While you take the lead for

Market Requirements Document (MRD)

Fig. 9 Typical MRD content

these documents, you'll engage other parts of the organization to benefit from their skill and insights.

As you're (hopefully) getting frequent new insights about your customers, competitors, and market, you may want to update the appropriate documents and re-release them from time to time.

MRD structure

There are many different MRD structures, and we describe some typical sections in the rest of this article. You will need to choose the ones that fit with your business.

"We invite existing customers and prospects to a workshop – we call these Customer Advisory Boards. This is to get their thoughts on what they expect from our software in detail, how they work, what their biggest frustrations are, and their expectations. Then when delivering a feature, we follow-up with them to show their opinion mattered (which naturally improves the customer relationship)."
Krisztina Orban, Product Lead , Al Tayyar Travel Group

Business objectives

This section should summarize why the company is developing the product or the next release. It may be to move into a new strategic market, grow revenue, or respond to a competitive threat. It identifies the business objectives and should show how what you're doing links to the company strategy and your roadmap. If a Business Case exists, it should be possible to copy across much of the content.

Target markets and proposition

The market opportunity needs to be fleshed out. This will include describing the segments you're targeting, the personas in those segments who'll value what you have, why will they care, how big the opportunity is and why you will succeed.

Market problems

The purpose of your product is to solve specific problems being experienced by your target market segments. So ideally, you need to describe what they are, explain when and where they happen, why they're important, and who or what is involved.

Stakeholders and users

Your product might impact many different people, and it's important to understand who they are and which requirements are important to them. They vary from product to product, but a stakeholder map (Fig. 9) will help you identify the ones relevant to you.

For example, the economic buyer at the customer cares about the business benefits the product will deliver. The technology buyer cares about how to implement and support the product. The users care about how it will make their life better and how easy is it to learn and use. So you need to understand which requirements will deliver these benefits.

MRD, PRD, BSD, FSD, HLD – the acronyms, content, and sequence of documents used to describe requirements and design is vast. Every company has a slightly different approach.

Also, don't forget about 3rd parties that might have requirements, e.g., a regulator or yourself as the Product Manager who needs a report on how the product is doing. An often overlooked one is your billing department, especially if there is any usage-based charging model.

In many cases, techniques like personas can be used to get insight into stakeholder and user requirements (see our Propositions Journal – Issue 2). One exception is the requirements for other systems, as these are usually precisely defined in interface specs.

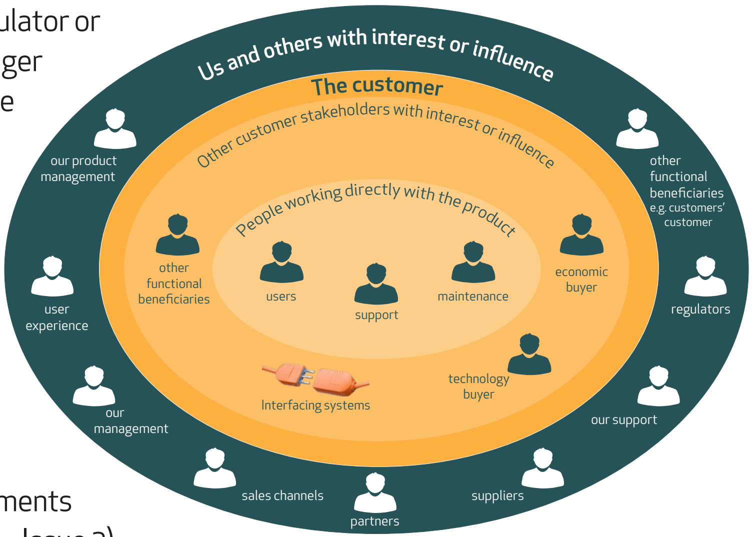

Us and others with interest or influence
The customer
Other customer stakeholders with interest or influence
People working directly with the product
our product management
other functional beneficiaries e.g. customers' customer
other functional beneficiaries
users
support
maintenance
economic buyer
user experience
regulators
Interfacing systems
technology buyer
our management
our support
sales channels
partners
suppliers

Fig. 10 Stakeholder map. Based on 'A taxonomy of Stakeholders, Human Roles in System Development' International Journal of Technology and Human Interaction Vol 1, 2005. **Ian F. Alexander**

High-level requirements

The MRD provides a high-level view of all the requirements. Doing so gives the reader a quick overview of the major things that are going to be needed to satisfy the market problem. These will often map to Epics or Themes in the Agile world.

To give some examples: for an enterprise SaaS (Software as a Service) product, the MRD might give a high-level view of the functions of a customer self-care portal, e.g., products covered, key functionality offered. An MRD for a consumer-focused, surround-sound amplifier might identify requirements for interfaces, audio and video standards, and design.

For software and hardware products, the MRD might also define the devices on which services should run, for example, dedicated hardware, web pages, apps, and TV.

Remember, these are market requirements, not product requirements, so they should all be rooted in the needs of the customers and link back to the market problem and what you will do to solve it.

"To ensure products address customer pain points, it is of paramount importance to capture proper VOC (Voice of Customer). While sales team feedback does bring insights, VOS (Voice of Sales) is not, and should not, be considered the same as VOC. Product managers should meet customers directly or at the very least on conference calls, to have open and transparent discussions, which will lead to meaningful product requirements."
Sophie Gigliotti, Global Product Management Consultant

Business process impacts

This section should summarise the impact on any business processes in the business, e.g., support, setup, billing. This can also help people understand the requirements in the previous section.

Key Process	Impact	Comment
Customer care support process	Low	Same 'help desk to help desk' support approach as with existing product, so minor training required and additional code in support tool.
Sales process and CRM	Low	Same direct sales model and process used, just need a new product code added for tracking. Reporting will be same structure as existing product.
Billing	High	Different pricing approach means likely that a new rating and charging capability will be needed.

Constraints

You need to be clear on the constraints under which you're operating. It may be a limited budget, the need to match a competitor product or to launch in time for a major event. Maybe you've already decided to use a 3rd party product, and you only need to cover how it will integrate with your existing systems and processes.

Context

The better the context you provide, the less room there is for ambiguity and uncertainty. Tools like scenarios and high-level stories can help. One approach is to create a set of product principles. For example, as easy to use as a TV, as reliable as Big Ben, can be used on the deck of a ship in a gale using gloves.

Another approach is to describe what the product is not. An enthusiastic developer can over-deliver against what was requested. Or they imagine a bigger future for the product and build in huge architectural changes to 'get ready' for that future. The cost for this is that timescales slip or something else doesn't get delivered. Documenting what the product isn't helps reign in this 'gold-plating' behavior.

Requirements management process

This is a description of how the detailed requirements will be gathered, prioritized, and communicated to the design and development team. It may be in a product backlog (Agile/Scrum), or a prioritized list. It should include a description of who does what and the documents and tools that will be used (e.g., a spreadsheet or a software tool such as Jira).

Conclusion

Documenting the vision, high-level requirements, and context for a product has a vital part to play in any development. It sets the scene, guides the development project, and provides the terms of reference for everyone involved. As a Product Manager who wants to build a successful product, we recommend you produce an MRD.

"A big thing that can go wrong with requirements is that context is lost. Somehow requirements get split up and set adrift from one another, and the developers go astray as a result of the inevitable slack in the interpretation of what has been written down. They deliver something that isn't fit for purpose or, more commonly, something that's just adequate when something amazing is within easy reach."
Nigel Shardlow, Director, Vanilla Internet Ltd

4 options

How to write functional requirements

"You want a $100m system by next month with no defined acceptance criteria. I can build it!!"
Richard Stevens (inventor of DOORS)

Requirements describe the details of a problem to someone who will create a solution. As product managers, they are the way we communicate market and customer needs or problems to the development and design teams with whom we work. What's interesting is that the end solution may not turn out as anticipated by the customer or us, but's that's fine if it solves the problem.

Writing requirements is hard work. It takes effort to understand what's really needed and more effort to write the details down. The way we write them has a big impact on how well they're understood and how much effort it takes to deliver them. When it goes wrong, we've seen it frustrate everyone involved, waste time, and deliver products that fail.

Classic

Simple structured language
e.g. The Product **Shall** support the following credit card validation

Typical words: **Shall, Should, May**

User Stories

User-centred with acceptance criteria

As a < user / stakeholder >
I want to < goal >
So that < benefit / reason >

Use Cases

Have a complex, pre-defined structure, exhaustive but time-consuming to write

Lots of sections e.g. roles, happy-day scenarios, exceptions, edge cases

Job Stories

Based on the jobs-to-be-done concept

When < trigger / situation >
I want to < motivations / forces >
So I can < expected outcome >

Fig. 11 Approaches to writing requirements

Which format to use?

Most of us just jump straight in and use the format we've used before or the one that's always used at our company. However, we've come across 4 different approaches to writing requirements over the past few years – the Classic approach, User Stories, Use Cases, and Job Stories. Each has their pros and cons.

Our view is that you need to choose the option that works best for your current situation. It will depend on the type of product you have, how closely you work with your Development/Engineering teams, what you are writing a requirement about, and the development process you use. The four options are shown in Fig. 11.

Classic approach

The Classic approach is to make a clear statement of what the product **shall do**. The aim is to keep the language precise and unambiguous. The intention is that once written, there should be sufficient information for the development team to build what's needed, with little requirement for additional discussion. Although it is likely, they will then take them and write another level of detail for each requirement!

Best practice is to write in the *active voice* rather than the *passive voice,* which tends to be clearer when users or entities are involved. For example, in the *passive voice*, you might say, "When the output state changes, it is logged in the event log." In the *active voice,* you would write, "When the output state changes, the system shall record the new state and the time of the stage change in the event log."

There are certain situations where the Classic approach is particularly effective. An example is when there are no users involved, and your product needs to integrate with another system that has a published Application Program Interface (API) with a detailed specification. The Classic approach often documents requirements to the same level as Use Cases, which are described later in this article.

"I'm sorry, I would have written a shorter letter, but I didn't have the time."
Blaise Pascal, French mathematician

User Stories

User Stories is the approach championed in Agile and, because of that, has become the one most commonly used – especially for software. The focus is on writing requirements from the point of view of someone using the product. This is known as 'user-centered requirements'.

The use of a standard structure coupled with everyday language makes them easy to discuss with customers, developers, testers, or anyone else. The structure also focuses effort on understanding the customer problem (the **what** and **why**). It helps stop the natural inclination when writing requirements of jumping to the **how** – which should be the responsibility of the design and development teams.

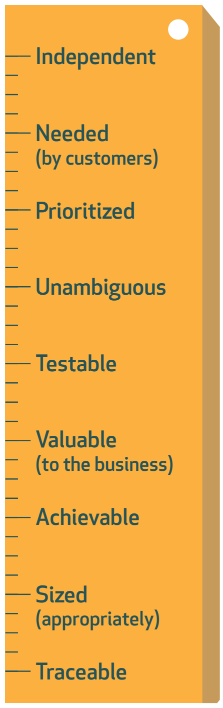


Fig. 12 Requirements should measure up to these criteria (INPUT VAST)

This type of requirement is often talked about at different levels. These are epics, user stories, and themes (Fig. 13).

"We've shifted to writing a 'Feature Brief' which sets the focus on a problem statement, feature benefits, evidence from the market,and perspectives from different personas which reinforce the need for the feature. It starts a conversation with the delivery teams and ensures solutions are discovered by the team building the software, rather than dictated by the product owners. Our software delivery teams are much happier now!"
Steve Brunt, Product Manager, IRESS

Epics

Epics describe the high-level requirements that are needed to solve the market/customer's problem. These requirements are too large to be built in a single iteration, so must be split into smaller user stories. Or, they may span the work of multiple development teams. It's useful to think of epics as large user stories, and they are often written in the same format. This is a good level for Product Managers to work at if they are not also the Product Owner.

Sharing the full set of epics gives everyone a view of what's seen as important and how the product is likely to evolve. When product investment is needed, then a full set of epics can be written to give a view of the complete optimum product. A rough estimate of the total development cost can then be made by a lead developer. With the iterative approach of Agile, things may change, but this gives a starting point to estimate costs and how long things will take.

If an epic is prioritized for development, it has to be broken down into user stories, which are discussed and estimated in detail in a sprint planning meeting. It's not worth spending time on lower-priority epics as they may never make it to the top of the list.

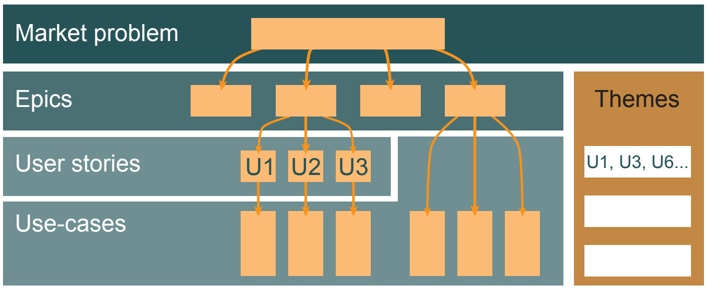


Fig. 13 Different levels of requirement

There may be several levels of decomposition to get to the point where the development team understand what's required, can produce a meaningful estimate of effort, and are confident a user story can be built within a sprint. If working in a Scrum environment, typically, the effort required should be less than half a sprint.

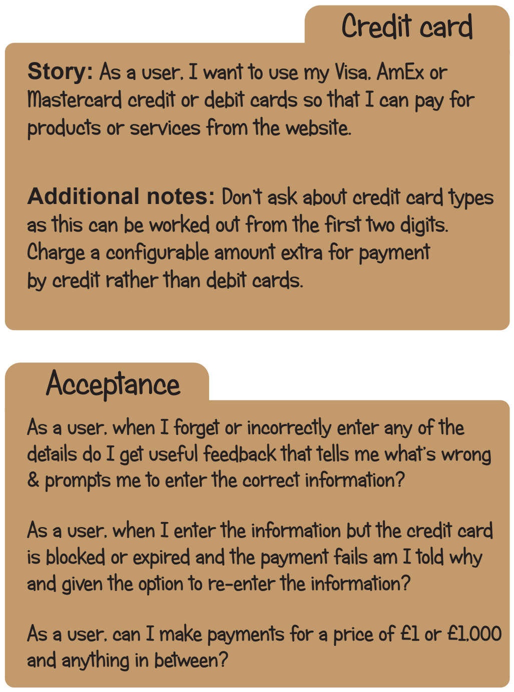
In Scrum, the user story is written by the Product Owner. The Product Owner needs to be someone with a high level of domain knowledge, so they can provide the detail needed by Development.

User story structure

As a <user/stakeholder>
I want to <goal>
So that <benefit/reason>

User stories should be kept as simple and as short as possible. In the world of Agile, the expectation from any local development team is that the user stories are the first stage in requirements definition.

User stories represent the requirements rather than providing an exhaustive description of them. The details are filled out through a **conversation with the development team** about the story and also by listing the **acceptance criteria** that must be fulfilled in order for the delivery of the story to have been completed. The additional notes section can also be a powerful area to capture any assumptions, constraints, or clarifications.



Credit card

Story: As a user, I want to use my Visa, AmEx or Mastercard credit or debit cards so that I can pay for products or services from the website.

Additional notes: Don't ask about credit card types as this can be worked out from the first two digits. Charge a configurable amount extra for payment by credit rather than debit cards.

Acceptance

As a user, when I forget or incorrectly enter any of the details do I get useful feedback that tells me what's wrong & prompts me to enter the correct information?

As a user, when I enter the information but the credit card is blocked or expired and the payment fails am I told why and given the option to re-enter the information?

As a user, can I make payments for a price of £1 or £1,000 and anything in between?

Fig. 14 User-centered requirements written as user stories

User stories and themes

Themes describe a collection of user stories that share a common attribute. For example, all the user stories associated with generating reports are grouped into a single 'reporting' theme. The implementation of themes might span multiple epics, and they're often used to help coordinate development activities and release planning. For example, a development team working on one part of the code gets all the user stories associated with a theme, or all the stories associated with a theme are delivered in the same release, so there's a strong marketing message.

Use cases

Use cases provide an exhaustive description of what's needed. They take a lot of work to produce but give a much more detailed description of the requirement.

A use case is more detailed than a user story (see Fig. 15) and may be written instead of, or in addition to, user stories. They are normally written by a Business Analyst or Developer after interviewing the customer and use their own jargon. The aim is to handle all possible cases so that Development can proceed without additional questions being necessary.

They're useful if it's going to be difficult to get ongoing clarifications from the customer or discuss things with Development. The drawback is that it shifts the responsibility for getting things right to the use case author.

Use-case

Functional goal: an active statement of the goal of the use-case.

Roles: identify the roles that are relevant to the use-case.

Happy day scenario: describe the steps of the normal use-case. This can be a text description, but it's often helpful to use a graphic to illustrate the flow of activities.

Variation scenarios: describe all the variations of what might happen as a series of scenarios, each with detailed steps.

Exceptions: walk through each exception scenario, the steps each entails and how they should be handled.

Preconditions: identify any preconditions to the use-case including the trigger that causes the use-case to happen.

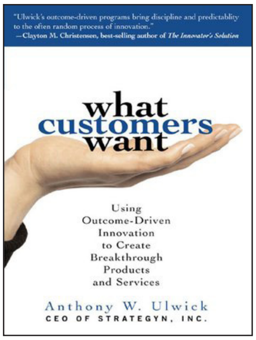
Completion: specify what must have happened in order for the use-case to be ended with successful and unsuccessful outcomes.

Stakeholders and interests: identify stakeholders and others, aside from the roles already identified, with an interest in the use-case.

Use-case specific qualities and constraints: identify any specific qualities or constraints, e.g., non-functional requirements that apply to this use-case (rather than those which apply to all use-cases).

Fig. 15 Use case format

Use cases are rarely used with Agile as they assume little ongoing communication and interaction with the development team (which is the very essence of Agile). But if you're not getting what's needed, you may need to resort to using cases. We've come across examples where companies started out using user stories but had to fall back on use cases because communication with the development team was so difficult. Language problems, teams split across different countries, teams working in different time zones, company politics, cultural differences, and the use of third parties can all have an impact.


"Ulwick's outcome-driven programs bring discipline and predictability to the often random process of innovation."
—Clayton M. Christensen, best-selling author of The Innovator's Solution
what customers want
Using Outcome-Driven Innovation to Create Breakthrough Products and Services
Anthony W. Ulwick
CEO OF STRATEGYN, INC.

What Customers Want by Anthony W. Ulwick

Job stories

This approach to writing requirements is relatively new and based on the jobs-to-be-done concept, which was popularised in Anthony Ulwick's book – What Customers Want. The approach was developed to address two concerns that some people have with user stories. Firstly, personas may be poorly defined or understood or simply irrelevant to a particular requirement. And secondly that a user doesn't sit around wanting every outcome or benefit all of the time, so it's often useful to understand the trigger for something.

There are different formats you can use, but a good example is

When a < trigger / situation >
I want to < motivation / forces >
So I can < expected outcome>

Job stories are a way of defining requirements by basing them on the jobs users are trying to do. The key is that the focus is on the event, not the user.

Here is an example from eBay.

"When a buyer has already made a bid on an item, they want to immediately receive a counter bid notification so they can have enough time to evaluate and update their bid."

This approach is good when we don't know much about the user or who they are is just not that relevant. What we care about is what they're trying to achieve. This is often the case with internet-based B2C services.

Summary

We've described 4 different ways to write requirements, and we know that others exist.

A cornerstone for good requirements is having agreement on their format. Your business needs to agree on the most appropriate format for your products and development environment. It should be what works most effectively and efficiently for you and the development team.

So, it makes sense to experiment with different approaches if you're not getting what you expect when you write your requirements. You need to pick the right tool to do the job.

"At the end of the day, there is no one-size-fits-all solution. If you're doing a technology refresh to drive efficiencies on an already successful product, then the focus should be on detailed non-functional and process requirements. For new applications to create the optimal customer journey, the requirements should work backward from the UX and combine job stories with business drivers. In-life products in continual iteration mode typically benefit from user stories governed by an agile methodology (Kanban, Scrum, etc.). You need to evaluate the best tool for the job at hand."
Dominic Carroll, Product Manager, NCC Group

More requirements

Non-functional, process, and UX

There are up to 4 different types of requirements that need to be considered to have a fully working product.

These are **functional requirements** (the features of the products – what it should do), **non-functional requirements** (operational characteristics of the product such as performance and scalability), **process requirements** (the workflows that involve people and which enable the product to be set up, operated and supported) and **customer and user experience requirements**.

The term non-functional requirements is a misnomer as it implies that it covers everything other than functional requirements. But it's one of those standard terms that's been around for a long time, so it is something you need to get used to!

These first 3 types of requirements deliver a 'working' product. They don't deliver a 'great' product. For that to happen, the fourth requirement is needed - this is aimed at delivering a great **customer and user experience**.

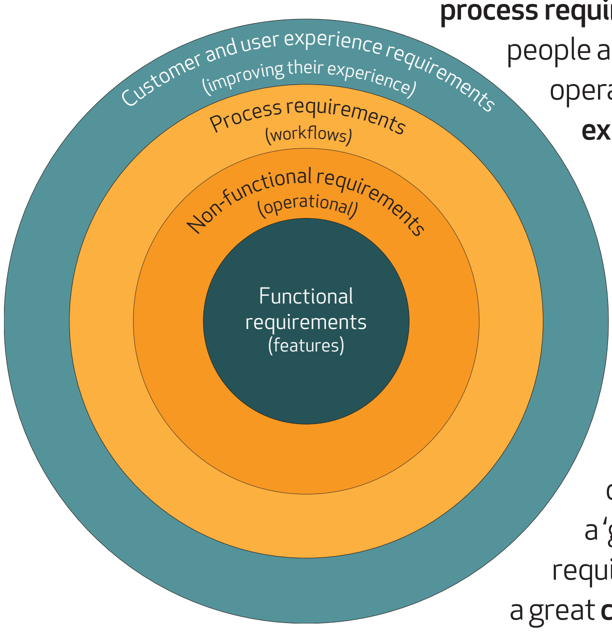

Customer and user experience requirements
(improving their experience)
Process requirements
(workflows)
Non-functional requirements
(operational)
Functional requirements
(features)

Fig. 16 Different types of requirement

Customer experience vs. User eXperience

Customer experience refers to the experience of the people in your customers (and prospects) when interacting with any of your company's people, systems and processes, e.g., to order products, get help or to pay for something. What's more often talked about is **User eXperience (UX)**. This relates to the perception and experience of your product by someone who's using it. In Business to Consumer (B2C) products, the user and customer are often a single person. In Business to Business (B2B), the user is often just one of many people that interact with your company and its products.

In defining functional, non-functional, and process requirements, the desired customer and user experience must also be considered and communicated.

At a basic level, customers are taking your product to achieve an outcome they value. For a B2C product, it might be to make the user feel great, e.g., look at me, I'm using the latest Apple iPhone. A B2B product user might want something that enables a tangible outcome, i.e., completing a step in a workflow as quickly as possible.

Giving a great experience can be a significant competitive edge. It can also reduce your costs for support, user training, and customer acquisition.

Use-case

Functional goal: an active statement of the goal of the use-case.
Roles: identify the roles that are relevant to the use-case.
Happy day scenario: describe the steps of the normal use-case. This can be a text description, but it's often helpful to use a graphic to illustrate the flow of activities.
Variation scenarios: describe all the variations of what might happen as a series of scenarios, each with detailed steps.
Exceptions: walk through each exception scenario, the steps each entails and how they should be handled.
Preconditions: identify any preconditions to the use-case including the trigger that causes the use-case to happen.
Completion: specify what must have happened in order for the use-case to be ended with successful and unsuccessful outcomes.
Stakeholders and interests: identify stakeholders and others, aside from the roles already identified, with an interest in the use-case.
Use-case specific qualities and constraints: identify any specific qualities or constraints, e.g., non-functional requirements that apply to this use-case (rather than those which apply to all use-cases).

Fig. 17 Some aspects of user experience

How is it done?

Ideally, the product will get the support of Customer Experience or User eXperience experts to help write an Experience Design Guide. This includes things like general design, screen layout principles, and branding guidelines. Another tool often used is customer journey mapping. It lets you walk through the various points of engagement between a customer and your company/product on their journey to achieve their objective and think about their experience at each stage.

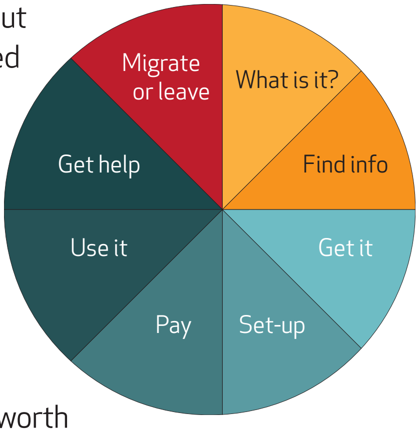


Fig. 18 The customer experience wheel

The customer experience wheel

A useful tool to think about different aspects of the experience is the customer experience wheel (Fig. 18). It's worth reviewing what can be done to improve the customer's experience at each stage of the wheel – especially if it's affecting the success of your product.

Story mapping

Story mapping is a simple technique to visualize the whole product.

There are many ways to do it, but one is to map user stories into a series of steps, i.e., the steps a user typically takes when buying and using a product. Under each step, you list the details (in user stories) that would be part of that step. It keeps the focus on the user and what they are trying to achieve. It also helps everyone to see the full picture, to identify ways to simplify things and to spot interdependencies between requirements.

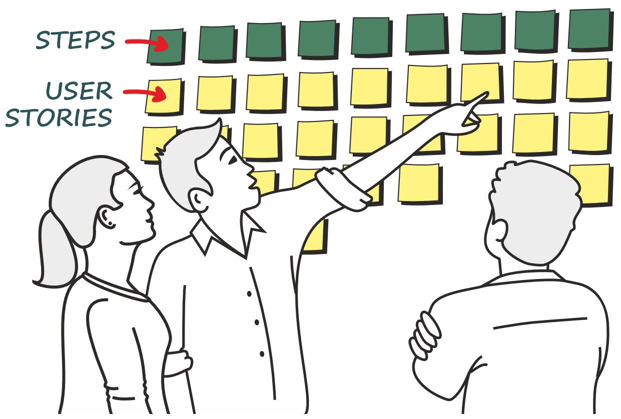


Fig.19 Using story mapping

"I tried story mapping with my Agile team. We huddled in place for two days while I described the high-level requirements. Then we walked through the user stories together to build a deeper understanding of what outcomes were needed. Afterward, one of the developers even told me that this was the first time they felt they truly knew what problem had to be solved."
Tyler Chapman, Product Manager, LexisNexis

Features are often easily copied

A further challenge is that we sometimes get drawn into spending all our time worrying about the features of our product and whether they meet market needs. But in a world where features are easily copied, it might be the strengths (or weaknesses) in our non-functional requirements, processes, or customer experience that are the deciding factor in our customers' decision to buy. As an example, one product I looked after was pretty similar to those of our competitors from a feature (functional) perspective. Still, we were successful because of our excellent customer support processes. In another company, we lost business despite having the best functionality because our customer experience was worse and our platform less reliable.

So, defining these non-functional, process, and customer and user experience requirements can be really important.

Non-functional requirements

Non-functional requirements describe the key performance or operational characteristics of the product. You can think of them as constraints on the product or its design. As with functional requirements, they can normally be written in different ways. If using user stories for your functional requirements, you can use this format, which gives consistency and helps understanding.

An example for performance might be … "**As a** user, **I want** 95% of my requests to be satisfied within 500ms and the information I'm given to be no more than 60s old **so that** I can get back to my customers without delay and with accurate information."

In B2B products, the Service Level Agreement (SLA) for your service may also need to be specified – for example, the time taken to respond to issues and the target time to fix them. These will guide the Service Level Objectives (SLOs) agreed with Operations or your outsourced partners.

Sometimes it can be powerful to describe these constraints in terms people are familiar with in their daily lives… "as a user, I want to find results in less time than it takes a coffee machine to make me a drink."

Requirement	**Brief examples**
Performance	User authentication in <0.1s
Security	2-factor authentication ISO 19092
Capacity	>100k transactions per second
Maintainability	Suports SNMP v3
Compatibility	Backward compatible 2 major releases
Legal	Compliant to BASEL III
Reliability	99.99% system uptime
Scalability	Scales to 100k users with no downtime
Localization	Import local language content
Accuracy	Log records accurate to <1s

Fig. 20 Examples of non-functional requirements

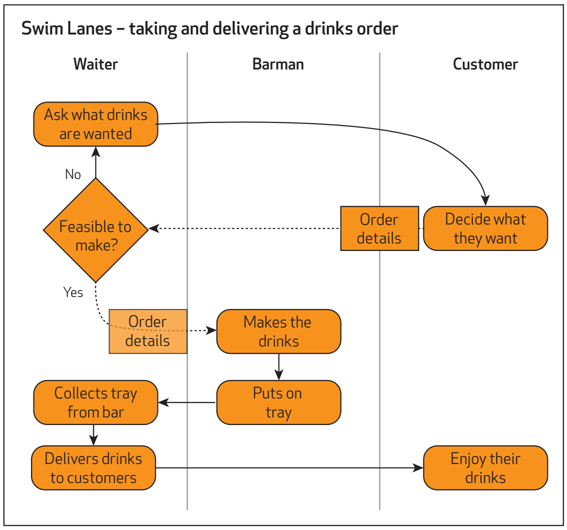
Process requirements

Processes define how your business operates. Sometimes these processes are fully automated, and sometimes they are heavily manual. For example, onboarding a small customer for an online product is likely to be done by a fully automated workflow that handles everything from initial sign-up to making the customer live. But, if, for example, your company delivers enterprise solutions to corporate clients, there is likely to be a mix of manual and automated processes. Examples of manual processes might include the completion of product setup and integration documentation or agreement for a support SLA.

For product managers of professional service products, e.g., consultancy, the processes and constraints are the key building blocks that define their products. They may have minimal functional and non-functional requirements apart from the need to be able to contract and bill for their services.

"When you get a requirement from a subject matter expert, always ask 'Why.' The question is key and may seem obvious, but asking can save tremendous effort. Quite often, what's actually required is very different, and you can save time and effort for other things."
Peter Klein, Industry Commentator

Poor processes and workflows can impact the quality of the customer experience or make your product expensive to run. If this happens, the product manager might prioritze the resolution of these problems in order to get their product back on track.



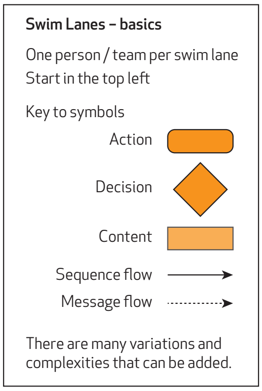


Fig. 21 A swim lane diagram illustrating a simple process

It's unlikely that a product manager will work on very detailed process design, but they often have to understand things at a high level. Swim lane diagrams are a great tool to visualize how things work and to identify improvements that can be made.

They show a flow of information, actions, and decisions within and between different teams or departments in an organization. Each team is represented by a different 'lane' in the swim lane diagram. Once you, or a business analyst, have created a swim lane diagram showing the current situation, then you can try and identify changes that might plug gaps, simplify things or speed things up. In the example above, it might mean each table has a menu of what drinks are available to order, so the waiter doesn't need to decide if a drink is feasible to make.

Summary

When you start thinking about a new product or product enhancement, it's usual to focus on the functional requirements, i.e., what the product needs to do. But remember, the non-functional process and customer/user experience requirements can make or break the success of your product.

Prioritizing

Tools to help you

We know prioritization is a big issue for many product managers as it comes up repeatedly in our annual survey. It's because every company is limited by the resources they have available to work on products, and the onus is on us, as product managers, to ensure that the company works on the most important things.

The prioritization challenge not only affects the products we develop but also our product marketing, design work process improvement, etc. – in fact, any product-related activity that requires company resources. And we face it whether we're managing a hardware, service, or software product.

A RHINO is a Really HIgh value New Opportunity. It's when a salesperson in a B2B company comes along with a deal for something you don't do – something that will trample all over your roadmap plans.
See. www.productfocus.com/blog/taming-the-rhino/

One might think that software products have prioritization cracked with their move to Agile development where re-prioritization is allowed every few weeks. But it's not always the case. We often see Development working on requirements for single customers or senior stakeholders at the expense of work planned for a market-oriented roadmap – work that later on, everyone agrees, was the wrong option.

So, we want to share some tools that can help – particularly those that can be used to prioritize product requirements. All use information on three key attributes – value, cost (or effort), and risk.

"Prioritizing requirements for an existing product is tough! Without new revenue streams, it may have no future, so you need some sizzle to attract new customers. However, the installed based is where the long-term loyalty and repeat business comes from, so you need to provide features to make them feel special as well."
Derek Britton, Director of Product Marketing, Micro Focus

Value, cost, and risk

Most prioritization comes down to a trade-off between the value something brings versus the cost or risk of delivering it. Some examples that can be used for value, as well as cost or risk, are shown in Fig. 21.

	Value examples
1	Revenue
2	Cost of delay
3	Aligned with release theme
4	Profit (e.g. Net Present Value)
5	Relief of customer pain
6	Validates learning
7	Essential to customer win
8	Relieves technical debt

	Cost or Risk examples
1	Cost (Dev, marketing...)
2	Technical risk (complexity, skills...)
3	Delivery risk (resources...)

Fig. 22 Options for how to measure what's important

In our experience, many companies make up their prioritization framework or adapt one of the techniques described below. They all use a scoring system, so you can rank ideas side by side and make a decision based on the score.

In the example in Fig. 23, we've used two factors for value – 'forecast revenue' and 'theme alignment' to what we have on our roadmap. Each is given a percentage weighting. Similarly, we've also used 'development cost' and 'delivery risk.' The typical formulae are Value/(Cost or Risk), and so you can see from the Priority Score of 1.9 that Candidate 4 would be the highest priority feature to develop.

	Value					Cost or Risk					Priority score
Feature	Revenue		Theme alignment		Score	Cost		Delivery risk		Score	Value / Cost or risk
	60% weighting		40% weighting			70% weighting		30% weighting			
Candidate 1	4	2.4	3	1.2	3.6	4	2.8	3	0.9	3.7	1.0
Candidate 2	2	1.2	2	0.8	2	5	3.5	4	1.2	4.7	0.4
Candidate 3	0	0.0	1	0.4	0.4	2	1.4	1	0.3	1.7	0.2
Candidate 4	5	3.0	5	2.0	5	3	2.1	2	0.6	2.7	1.9

Fig. 23 An approach to calculating priorities

Cost of delay

Cost of delay is a way of assessing the value or cost of something relative to the date at which it's delivered. An ice cream freshly bought on a hot day is a valuable thing. Wait 30 minutes before you eat it, and there is nothing left but a mess on your hand – the value is considerably less. So, to make good decisions, we need to understand not just how valuable something is but how urgent it is. The cost of delay is normally measured as a cost per week or cost per month, e.g., if we deliver this requirement after this date.,the cost of delay is €10k per month.

RICE (Reach, Impact, Confidence, Effort)

This is a method used and popularized by Intercom, focused on prioritizing features in a mass-market online product.

Reach: how many people will this impact? (Estimated within a defined time period e.g. a quarter)

Impact: how much will this impact each person? (Massive = 3x, High = 2x, Medium = 1x, Low = 0.5x, Minimal = 0.25x)

Confidence: how confident are you in your estimates? (High = 100%, Medium = 80%, Low = 50%)

Effort: how many 'person-months' will this take? (Using whole numbers so as not to get 'into the weeds' of estimation)

RICE score = (Reach x Impact x Confidence)/Effort

Weighted Shortest Job First (WSJF)

A prioritization method promoted as part of SAFe (Scaled Agile Framework). It uses the information on the cost of delay and the amount of effort to deliver each requirement (job) to work out the ideal sequence in which they should be delivered. The sequence aims to deliver the highest value as fast as possible.

A standard process to reduce arguments

Having a standard process makes it easier to justify and defend your decisions. If you've agreed on the process in advance with your management team, the hope is that the logic used can help avoid major arguments about priorities, e.g., when a RHINO comes along. When a Senior Exec or Sales Manager says, "Can we just add it to the roadmap?" you can say, "That sounds like an interesting idea – let's evaluate it against our agreed prioritization criteria and see how it scores compared to everything else we have planned." It can be even more powerful if you give them ownership of inputting the data that drives it as they feel like they have more control.

"We score portfolio level requirements based on how well they address our key business objectives. These are weighted accordingly and pre-agreed with internal stakeholders. This gives us an objective measure of why one piece of work is higher up the strategic backlog than another."
Daniel Noble, Product Manager, ECi Software Solutions

Practical considerations

There will always be some element of subjectivity when scoring things. We recommend working closely with key stakeholders from areas such as Development to improve the accuracy of the scoring. Also, keeping things simple – having fewer criteria in your priority calculation gives people more confidence in the process.

There is also a danger that people will try to 'game' the system. So try to predefine the criteria in collaboration with the other teams to reduce any ambiguity.

Consider constraints

Regardless of which approach you've used to define your priorities, there's often an extra step that you must take, i.e., considering constraints that impact on your ability to build something. For example, critical people or infrastructure might not be available. Dependencies with other work might be unavoidable, and market timing might impact on whether something should be pursued. Having insight into these constraints will help fine-tune your priorities.

"Ask a single customer, and you will get that individual's blinkered view. They may be expert, but they don't represent all your customers, no matter how friendly you are with them. Sometimes it's better to work with customers that are not your best friends, you're more likely to hear the truth, and you might win them round."
Colin Hamp, Product Launch Manager, CDK Global (UK) Ltd

Valuing all input

Let's be honest, not everyone's input carries equal weight. A requirement from a key customer who's a great advocate or highly profitable is more important than one from whom you make little money. But you have to manage expectations carefully to avoid alienating customers whose requirements you don't take forward.

We have to be able to answer 3 questions to know if a requirement should be taken seriously: Who values this requirement and why? Why do we care about their opinion? And What's the value to the business?

The CEO's opinion will always be important, and your job may depend on how you handle their input! You'll need strong evidence and reasons to argue against their wishes. This is where the tools above can help by showing how you've made your decisions based on rational arguments.

The temptation to have too many requirements

When you're considering development work that spans many months or years, such as with an Agile release train in SAFe or a big Waterfall project, you might fall foul of the temptation to overload Development.

One reason this happens is that no one really knows what will be important that far in the future. As a consequence, customers and product managers ask for everything they can think of – just in case!

A second reason is that we try and influence things over which we have little control. The unfortunate truth is that there are limits to how much influence we have over the full scope of the proposition a customer buys.

When a customer takes our product, what they are buying is often a mix of stuff that is product-specific (e.g., its functionality, pricing) and stuff that is common to many of our company's products (e.g., the brand, repair, and support processes).

You might have a lot of control over those things that are specific to your product but little influence on those capabilities that are used more broadly. There are other areas where you might have an interest in what's going on, e.g., support processes, but little influence or control. While you can and should try to expand your scope of influence, you need to focus effort on those areas where you can make a difference.

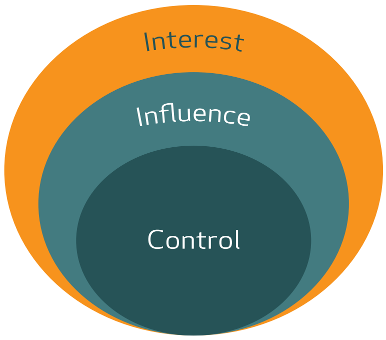


Fig. 24 Varying degrees of control

Conclusion

Tools like these help us make defendable and robust decisions on the priority of requirements. Without this, chaos reigns as each stakeholder argues louder and louder for their vested interests. As product managers, we must make these tough priority decisions to ensure the company's resources are not wasted, and we develop products that customers want and that also make money for our business.

"Requirements prioritization is a hugely political balancing act in many companies. Everyone from Sales, to Support, to Engineering, to Management, and of course, Product Management have an interest and agenda. To minimize politics, ensure all parties are aware of the trade-offs that are being made, both in their favor AND against them – and why. The 'why' is critical, as it gives context for the decisions. Most people will accept some compromises as long as they understand the reasons."
Saeed Khan, Founder, TransformationLabs.io

Tools to use with customers to understand priorities

Getting direct input from customers to help prioritize requirements is very powerful.

Kano analysis is a great approach to working out the features for your next product release. They are divided into 'must-haves' that customers really expect to be there, 'valued features' where the better they are, the more customers will pay, and 'exciters' that aren't expected, but that really get customers talking.

To undertake Kano analysis, you need a candidate list of features and customers to give their opinion on how well each feature matches their requirements. Sometimes internal teams do this based on their understanding of what will matter to customers. The output can be shown in a table or on a graph. Ideally, your next release needs to include all the must-have features, one or two exciters to get customers interested, and as many of the valued features as can fit.

Outcome-Driven Innovation (ODI) explores the jobs your customers and prospects undertake (the tasks they're trying to achieve). This aim is to understand where they're being let down by their current solution. Its strength lies in its focus on what customers care about, i.e., the outcomes they need to deliver in order to do a good job. It provides insight into what these are, how important they are to customers, and how satisfied they are with their current solutions. With that insight, you're well placed to prioritize what they'd value and come up with potential solutions. See Fig. 25.

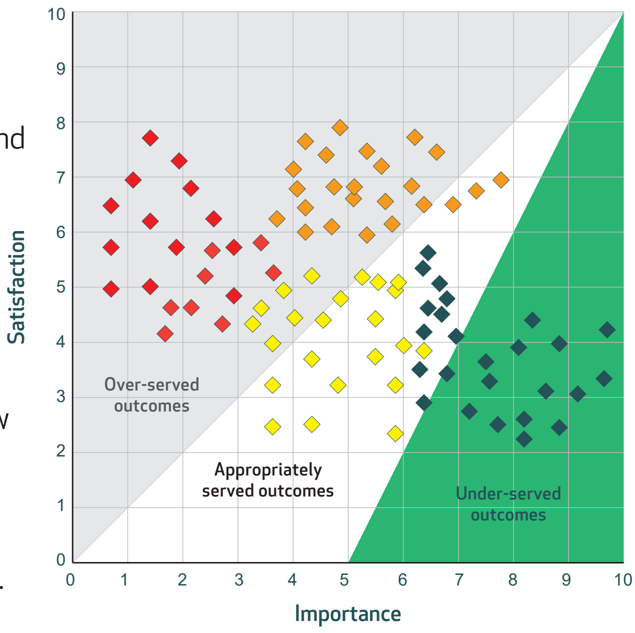


Fig. 25 Using ODI to understand what matters to customers

Innovation Games are a great tool for a discussion with customers about their priorities. One of these games, Product Box, gets customers to sell your product back to you. Understandably they tend to focus on the things they really value. They're fun and engaging, which means you get customers enthused and talking about what they like, don't like, and want from your product.

Finished

When are you done?

Have you ever had the situation where Development is keen to move onto other priorities after saying they've finished delivering your requirements and you're thinking – no, you haven't!

Knowing when things are complete can be a source of frustration in the requirements process and also a cause of product delays.

So how do you know that a requirement has been delivered?

Acceptance tests

The most obvious things that need to be considered are acceptance tests. All will need to be passed for the requirement to have been delivered.

If your requirements are written as user stories or job stories, then the criteria for these are often written by the Product Manager or Product Owner. They'll be further refined with the Development or Quality Assurance (QA) team. On average, 4 to 5 acceptance tests is typical (though we've seen up to 100!).

"As far as Done, sometimes it's never done its just done for now or close enough."
Duncan Rodger, Product Manager, Bentley

How much detail?

Good acceptance criteria and tests are a critical tool in ensuring user stories result in a product that delivers against business requirements. We see many companies with a standard way of writing acceptance criteria – the method that was taught on the 2-day Scrum course employees attended. However, we think you should be as exhaustive as you need to be. If you work with a team of developers who are inexperienced on your product or to which you have limited access, then it's often better to be more detailed in your acceptance criteria to ensure you get what you want. When they start delivering reliably, then you can ramp down the detail you provide.

If you're using classic requirements or use cases, then it's usually a Business Analyst who will write the acceptance criteria, with a QA engineer/tester turning them into tests.

The key is to be clear on what you need the product to deliver to customers to demonstrate the business requirement is satisfied.

Acceptance criteria formats

Acceptance criteria can be written in many different formats – 4 are shown in Fig. 26. The first uses questions, the next two use statements, and the last one uses tables to show expected behavior in edge cases. Two of these approaches (the first and third) use a standard structure (as shown by the bold text), which are popular as they help the testers create tests and add them to automated test tools.

Questions

Example acceptance test for a service enabling users to store information about websites they visit

As a consumer end-user, **can I** choose sites for which visits should not be recorded?

As a consumer end-user, **can I** see a list of the sites visited and when they were visited?

As a consumer end-user, **can I** delete visit details for site visits made within a date and time range that I specify?

…

Statements - 1

Example acceptance test for a service that allows administrators to grant user access to a system

If I am an Administrator, I can create User Accounts.

I can create a User Account by entering the following information… [add info into test]

I cannot assign a new User to report to an 'Inactive' User

…

Statements - 2

Example acceptance test for a service that allows administrators to grant user access to a system

Given that I have granted access to the system for a user **when** they access it. **then** an entry of their login time is created in the log

Given that I manage user access **when** I add a user, **then** they and I are emailed with their login details

…

Tables

Example acceptance test for a service holding pre-payment that is being used to pay for tickets

Funds held	Ticket cost	Payment status	Final balance
€ 50	€ 40	Approve	€ 10
€ 30	€ 40	Reject	€ 30

Fig. 26 Different styles of acceptance tests

However, acceptance criteria are not the only things that need to be complete.

Aligned to guidelines

We would expect a demonstration, not just that acceptance criteria for individual features have been fulfilled, but also that architectural and design guides have been followed. The author had a painful experience with this when nine months of Agile development work was binned because architectural guidelines hadn't been followed.

"In my experience, when a developer states that he's done, usually he's not completely done yet (only one little thing to fix). Discussing a definition of done for a sprint or a release upfront will prevent annoyances at the end."
Irmgard Doremans, Solution Architect, Prodware

Technical debt (with software)

Taking technical shortcuts often seems attractive at the time but creates 'technical debt' that needs to be paid back at some point by rebuilding or refactoring software. The business needs to track whether the technical debt burden has increased because of the development. This is normally the job of Development, but it's worth asking the question – how is this tracked, and how will it be resolved?

Documentation

A problem we see in many businesses is a lack of documentation – when the developers moved onto something else, no one knows just what was built. That means Product and Marketing teams risk mis-selling the product and Support struggle because they don't know how the product is supposed to work. It can also make ongoing maintenance and enhancement much harder.

We would expect release notes to be written by Development with the requirement not considered as done until they are in place.

User-facing documentation – whether online, built into the product, or printed, also needs to be created. While input is needed from Development, they're usually not the best people to do this. It's often the responsibility of product teams. If you have the budget, then consider outsourcing this to technical writers who can do it quickly, clearly, and competently.

Releasable

And finally, is what's been built releasable? You may want to insist that the requirement isn't complete until all the processes and workflows are in place so you know how you can provide the new product to customers.

The Review

Reviews of great books for product managers

Book Review

Jeff Gothelf

LEAN VS. AGILE VS. DESIGN THINKING

What you really need to know to build high-performing digital product teams

SENSE & RESPOND PRESS

Principles for the Next Century of Work

Jeff Gothelf & Josh Seiden
Series Editors

Lean vs Agile vs Design Thinking by Jeff Gothelf

“At the end of the day, your customers don’t care whether you practice Agile, Lean, or Design Thinking. They care about great products and services that solve meaningful problems for them in effective ways.”
Jeff Gothelf

A great book about the potential conflict between 3 current product creation approaches that are all seen as best practice in the software world. These are Lean with the idea of Minimum Viable Products (MVPs). Agile with the rituals and language of Scrum, Kanban, and others. Design Thinking with its focus on customer empathy and thoroughly understanding customer problems.

It gives a short primer chapter on Lean, Agile, and Design Thinking and finishes with 10 recommendations on key practices to get the most out of them all.

In the introduction, Jeff uses an example of a client he worked with that had the tech teams learning Agile, the product teams learning Lean, and the design teams learning Design Thinking. The client expected these to all knit nicely together but found that each team had different working cadences, practices, and measures of success.

The engineering teams were focused on shipping bug-free code in regular release cycles (sprints). Product managers were more interested in learning with MVPs and driving efficiency through tactical backlog prioritization. Not to be left out, the designers sought to bring the customer front and center by validating the problem-solution fit with Design Thinking, but their up-front research and design exercises were seen as too lengthy and an unnecessary delay to product launch.

“Each discipline was working through its own ceremonies and tactics, targeting an ideal state of success unique to them. These different ways of working meant that the collaboration, shared understanding, and increased productivity they were all promised was nowhere to be found.”

Now we don’t agree with the focus that the client’s Product Managers had – to us, this sounds more like the Product Owner role. We advocate a wider and more strategic role. However, we do think this book is a great read, and at only 42 pages, it doesn’t take long!

What's your view?

Thank you to everyone who submitted quotes for this Journal. We're sorry there is no room for them all. Some of the best are shown below.

"I've always found that good requirements come from the product manager having empathy; firstly with the target users and their needs, context, and habits to ensure you're solving the right problem to begin with, and secondly with the designers, developers, and testers whose job it will be to make your product vision a reality."

Jock Busuttil, Product Management Consultant, and Author, productpeo.pl

"For me, the question is not just 'What are the customer requirements?' but 'What are the customer problems?' Be careful when you tell customers your ideas as this may bias discussions away from the real customer need."

Stephen Speirs, Senior Mgr Services Product Mgmt, Cisco Systems

"We find detailed use-cases work best when the development team is distributed. Storyboarding works best when the team is co-located – everybody hears the story, can translate it well, and be more agile in responding to change."

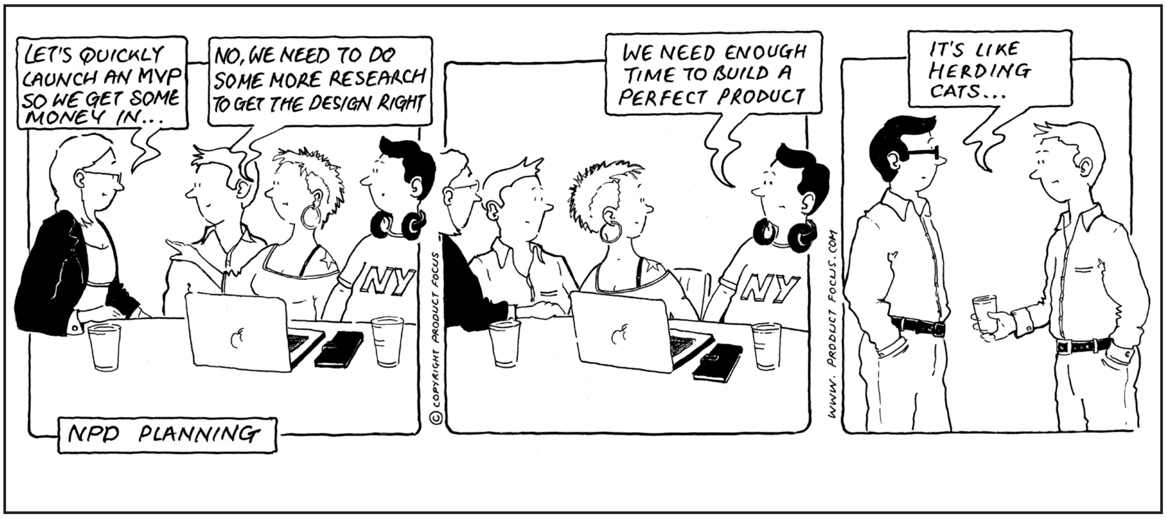
Ramachandran Sekaran, VP, Technology Solutions, Europe, Iron Mountain

"The problem with requirements is that developers don't read them. They don't even look at them until they are about to work on that piece of functionality. That's why talking, and white-boarding matters, and why properly managed Agile development works where *waterfall* doesn't."

Fred Smith, Product Director Information and Tracking Services, International SOS

Annual Survey

We use our survey to benchmark product management each year. Let us know if you'd like to take part. You can **download the latest results** from our website.

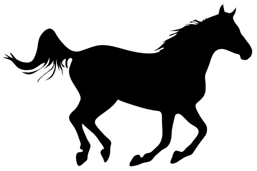

LET'S QUICKLY LAUNCH AN MVP SO WE GET SOME MONEY IN...
NO, WE NEED TO DO SOME MORE RESEARCH TO GET THE DESIGN RIGHT
NPD PLANNING
© COPYRIGHT PRODUCT FOCUS
WE NEED ENOUGH TIME TO BUILD A PERFECT PRODUCT
WWW. PRODUCT FOCUS.COM
IT'S LIKE HERDING CATS...

The Insight

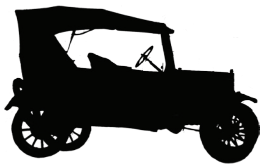
Go deeper – the problem is the problem

With requirements, the big question is not 'What does the customer want?' but What are the customer problems?'

When Steve Jobs was asked, 'What market research did you do that led to the iPad?' he replied, "None – it's not the customer's job to know what they want." Although not entirely true (there were years of prototypes), it makes a point. It's similar to the cliché that is often attributed to Henry Ford "If I had asked people what they wanted, they would have said a faster horse!"



vs



Asking customers to detail their requirements is an obvious place to start, but there are some major drawbacks.

Their answers will be based on an evolution of what's already there. They will describe the solution they think they want, but the underlying problem may remain unresolved. Do you really want faster mobile internet, or do you just want it to be more reliable?

In addition, customers expect us to be an expert on what's possible. They want us to come to them with options and help them choose between them. We can't really do that without thoroughly understanding **why** they want what they want.

"People don't want to buy a quarter-inch drill. They want a quarter-inch hole."
Theodore Levitt, Harvard Professor of Marketing

We can get that from open and honest conversations along the lines of What are your current challenges? Why are they a problem? Can you give me specific examples? We can also observe how they do things to identify inefficiencies and workarounds our product could solve.

These deep insights help us make the right decisions when it comes to prioritizing the requirements that will make a real difference.

So, although asking customers what they want is a great start, input from sales is important, and your backlog of requirements very useful – it's getting a deep understanding of customer problems that really counts. That means talking to them face-to-face. The answers you need are outside the building.

Training and Support
for product management leaders

Learn how to manage a product management function and team

Start up or improve product management with a Product Focus Review

Explain the value of PM with an Executive Briefing

product

focus

www.productfocus.com

product

focus

Learn best practice and improve performance with the European leaders

If you'd like to discuss product management training, or how we can support your product management function, please contact us:

+44 (0) 207 099 5567

info@productfocus.com

www.productfocus.com