Why Agile Marketing is the Key to Successful Products

Can Agile Marketing give you an edge?

The past five years have witnessed a sea change in marketing methodologies and processes at a rate that continues to accelerate.

Marketers have always been under pressure to demonstrate results with scarce resources, but now those results must be driven by marketing automation and validated by supporting metrics.

Buying audiences are increasingly fragmented, requiring a more tailored approach to content, messaging, and outreach which is being addressed by new approaches like account-based marketing. Social media and digital marketing technology have marketers constantly seeking ways to be more nimble and responsive, and have fundamentally changed the way they are going to market.

The days of executing static “big bang” campaigns planned months in advance are gone. Do more with less, and do it faster!

That’s the message that software programmers heard over a decade ago, and the creative way they responded has completely transformed how software is developed.

Agile software developers take incremental steps, see what works and what doesn’t, get team feedback, then move onto the next phase of development.

Their close collaborators in product management were quick to adopt the new “Agile” method, which is an iterative approach to product development that values adaptability and responsiveness to change as opposed to long-term and somewhat inflexible planning.

Agile innovation history pre-dates software development (you can read about it here).

Many experts believe that the Agile approach will expand well beyond IT. The spread of agile raises intriguing possibilities.

What if a company could achieve positive returns with 50% more of its new-product introductions? What if marketing programs could generate 40% more customer inquiries? In today’s world of constant buyer feedback, marketers must be able to adjust their plans based on changing market dynamics and what’s trending in digital and social media.

They need a better way to manage their marketing projects to achieve success, and that’s why many are considering Agile marketing frameworks.

For a large proportion of product marketers, Agile is relatively new.

According to the Wrike 2016 State of Agile Marketing report, 34% of marketers have implemented Agile within the last 6 months, and another 22% are just starting.

Product marketing teams see several benefits associated with Agile methods. Improved quality of work, speed of execution, and alignment on priorities are the top three benefits identified.

Marketers also see Agile providing them with more visibility into the overall project status, as well as helping them to more rapidly identify roadblocks, scheduling issues, or other problems.

Delivering experimental campaigns, getting feedback, iterating, and then building on the winning concept is how marketers are achieving success today.

By following the methods established by software developers and adapting them to the marketing environment, product marketers can explore and discover new ways to reach their buying audience and achieve their business goals.

Agile marketing takes time to learn and adopt, but eventually becomes a mindset.

For marketers who practice agile marketing, customer experience is the focus of everything: solving buyer problems and the buyer's journey, not on selling and the sales cycle.

So, what are the core traits of an Agile marketing campaign?

Rapid response and adaptation to changes in the market – You must still plan, but you aren't locked into it when the situation changes (and it will).
Multiple, rapid campaign iterations that can be tested and fine-tuned over time – There is less risk and more upside, since you are not placing your bets on a single, untested campaign.
Validation of choices using hard data (vs. intuition) – Leverage the wide availability of apps and services that capture and deliver marketing metrics and analysis.
Provision of individual, customized buying experiences vs. one-size fits all – Adopt an account-based marketing approach that addresses the specific need(s) of the buyer.
Cross-functional collaboration with team members in the organization and elimination of silos – Ensure messaging consistency as well as a positive user experience for the buyer. In the mind of the buyer, their interaction is with your company…not the sales, PR, finance and marketing departments.
Once implemented, Agile can help marketing departments, large and small, be more efficient, serve their customers better, and be more closely aligned with the business and sales goals of an organization.

Activities can be accomplished more quickly and with greater likelihood of success while maintaining a high level of transparency and accountability.

Agile Marketing Using Scrum

Many people in the technology world mistakenly believe that Agile is something new, but it really isn't.

Back in the 1940's, Toyota adopted Agile manufacturing methodologies (specifically Kanban) to reduce waste and increase transparency across teams, and as a way to promptly address constantly changing customer needs.

Agile Marketing With Development

Using Agile innovation for software development may help accelerate and optimize completion of a product. But if marketing is not involved in the process, then developers simply run a higher risk of more quickly creating a product that is not a "must-have".

With Agile, teams can more efficiently develop software that is released on-schedule and without bugs. However, if the product is not viewed as a necessity in the marketplace, then the outcome of the development process cannot be described as a success.

According to the Standish Group's 2015 Chaos Report, only nine percent of software projects at large companies are successful and for every 100 projects started, 94 will have to be re-started at least once.

The primary reason for failure was a lack of customer input and involvement.

Too often, marketing is not a part of the Agile development conversation.

The resulting product development process can sometimes be described by the phrase, "Ready, fire, aim", as the development team forges ahead without a clear understanding of market requirements.

For Agile to succeed… for the development team, marketing team, and the entire company…the process must include capturing, validating and clearly defining buyer requirements for the product before and during development.

That means including marketing in the process.

Agile marketing is an iterative and experimental approach to marketing that values adaptability and responsiveness to change over inflexible long-term planning, as well as two-way marketing interactions and collaboration among the various marketing disciplines. It builds on the agile methodology that has been widely adopted by software developers since the early 2000's and has transformed how software products are created.

Agile Marketing Using Scrum

There are several Agile models that marketers can adopt to not only improve overall marketing efficiency, but can help them to be more in sync with the software development teams that can benefit from marketing collaboration.

The Agile marketing method I am most familiar with is Scrum, as it has been the preferred tool of most of the development and product management teams I have worked with. (Another popular Agile method is Kanban). Agile marketing can use Scrum to manage their work, just as Agile development teams use it to manage software development.

Within the context of Scrum, marketers break down tasks within a project into digestible portions that can be accomplished within timeframes between a day and a week long.

The goal is to clearly understand and communicate what needs to be completed, what the market driver is, and who is the recipient of the output.

Scrum revolves around a regular series of events, and has its own “Agile-speak”.

A scrum sprint is the period of time the team has to complete their current projects. This can be from two to four weeks, depending on the complexity of tasks. Sometimes it’s a good idea to break up complex products into multiple sprints, spaced two weeks apart. That way you have a regular opportunity to check on status and make adjustments as needed.
A stand-up meeting is a daily scrum event that includes a project’s entire agile marketing team. These meetings are brief and to the point, with each team member concisely describing yesterday’s accomplishments, what’s on deck today, and problems they may have encountered. It’s an opportunity to identify and address any roadblocks and clear the way for progress.
In a sprint planning scrum session, the agile marketing team meets with sales and executive management for an update on the latest sales and corporate priorities. This is the time to fine tune activities and make any necessary changes and additions to the marketing backlog. The team then prioritizes and accepts enough work from the backlog to fill the next sprint.
A scrum sprint review occurs when a sprint is completed. It’s where the marketing team reviews the results of their efforts with their peers, and is analogous to the sprint demo that software developers have at the end of their sprints. The sprint review is an opportunity for marketing to share what they have been doing with sales and executive management, and provides a forum for accountability.
The follow-up scrum sprint retrospective is an agile marketing only meeting, where the team can assess the sprint’s successes and failures, and make appropriate adjustments for the next sprint.
This chain of events is repeated throughout the life-cycle of an agile marketing project using scrum.

Agile innovation process should always begin with a true understanding of your market and buyer, and subsequently translating that into a meaningful set of product requirements.

The key to product success is not limited to integrating product development processes, like planning, coding, testing and QA. It must also include ongoing collaboration with marketing with an ear to the voice of the customer.

Five Product Manager Resolutions (and a Little Humor) for the New Year

1. Use data and examples to further establish yourself in your company as the voice of the customer and the most credible expert on the market and competition.

As a Product Manager you should be the defacto authority on both your product and its market. No one should know more about your customers and market than you. If your product is a USB drive, you better eat, sleep and breath USB drives (not literally, do not try this at home as it may be a choking hazard).

Remember that you are the representative for your customer so have an accurate idea of what it is they really want.

To truly be seen as credible, it's important to have real numbers backing up your claims (even though 83 studies show that made up numbers can be just as useful as real statistics). The more often you use data, the more credible you will be.

2. Be terse in all communications – keep them short and effective – make your point and move on

Product Managers are always busy, it's just a fact of life; so are the people they are communicating with.

For that reason, it is generally not advisable to engage in too much chit chat or use flowery, imaginative language like you're writing a fantasy novel; just say what needs to be said.

In college, I (my parents, really) spent well over one hundred thousand dollars so I could learn about the 3 Cs of business writing: Concrete, Concise, and one more which I have since forgotten.

When you have a critical job to do, don't be afraid to be frank and straight-forward in your communication and other people will probably appreciate it unless they're a real jerk or a crybaby.

3. Build and deepen five key relationships that are critical to your success in your current job and to advancing your career in the future

Fostering relationships is crucial for Product Managers since they are working in a cross functional team with people of various backgrounds and they generally have no formal authority to boss anyone around.

Starting a new relationship and/or strengthening current relationships can seriously aide your current success and help you climb the corporate ladder. Often times it's not just about how good you are, it's about who you know.

Do favors for people. Take them to lunch. Buy your team donuts (this may not help their new years resolutions but it will help yours).

4. Get trained and get a great set of templates

Nobody is born knowing how to do a feature prioritization matrix or make a product and strategy roadmap.

Interestingly very, very few people are actually trained on how to be a Product Manager (less than 2% of those doing the job) and most of them are just thrown in the deep end and learn how to swim that way.

To continue with the swimming metaphor, that will probably work in the sense that you won't drown; but you almost certainly will not be able to swim as effectively or efficiently as someone who has taken lessons.

The same idea is true in most walks of life and Product Management is no different. Going through specialized and excellent training such as the Optimal Product Management and Product Marketing or People Skills for Product Managers courses will allow you to be a far superior Product Manager.

Additionally, using a collection of best-practice templates like the Product Management Office will ensure that you can create better work, using less time and resources.

5. Have more fun and make work more fun for your team and peers. A wise man once wrote that if you didn't do something fun for others at work today then your day was a failure.

This may seem to contradict with #2 a bit in that you might be wasting people's time, but it doesn't have to.

Just because you take your work seriously and get the job done does not mean you can't also have a lot of fun.

Make jokes (when appropriate), get to know your coworkers better, share meals, organize fun activities for your team, generally try to have a positive attitude.

You spend a LOT of time at work and with your work team so you may as well try to have some fun with it. These years keep on passing and after all, life is meant to be lived and enjoyed.

The Myth of the Iron Triangle

I am always amazed at how often accepted truths turn out to be false.

I am further astounded at how liberating it is to throw off the constraints imposed by those beliefs. The first myth that Kent Beck, the father of Extreme Programming, shattered for me is that change does not have to be expensive. The cost of change can remain relatively flat throughout all phases of the development lifecycle by re-organizing and re-thinking how software is developed.

The second myth, which I am now ready to castoff, is the Iron Triangle.

The Iron Triangle (figure 1) influenced much of my career and decision making.

It defines the relationship between scope, resources, and time. I was taught that you could not get more of one without more of the other. Thus, if I wanted to add more scope to my release, I would have to add more resources, time, or both. I could also fix one and then try to optimize the other two. But the relationship between these three qualities was defined and rigid.

There was one backdoor. You could play with the sacred 4th dimension of the triangle – Quality (Yes, I recognize that this would then be a quadrilateral.). No one wanted to deliberately state that we would compromise quality, so it was seen as fixed. But this is where the model and reality often diverged. Projects often sacrificed quality to add more scope and/or deliver faster. Lowering quality was the one way to create an exception to the Iron Triangle.

Agile Product Management Iron Triangle
Figure 1: The Iron Triangle

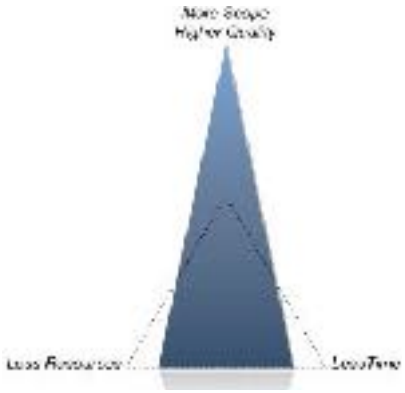
But with test driven development in an Agile environment, teams do not defer defects. Quality is built in.

Because less time is spent fixing bugs, the team is more productive, time to release decreases, and more scope is completed (figure 2).

Teams transitioning to Agile describe this effect as going hyper-productive, where they enjoy throughput gains of 100% – 200%. I have also observed this firsthand. From my experience, I concluded that scope and quality were compatible goals and positively correlated. But I still

didn't really question the Iron Triangle. It wasn't until I viewed this problem through the Lean perspective that the fallacy became apparent.

Agile Product Management Iron Triangle Less Time and Resources
Figure 2: The new reality



Lean looks to create fast, flexible, flow (see "Lean Thinking" by Womack and Jones):

1 Flow value (e.g. products, features, enhancements) through the development system
2 Do this in the least amount of time
3 Only create a new feature or an enhancement when a validated customer needs emerges

A system that achieves the above would be considered "perfect." Further, Lean teams spend time analyzing their processes to uncover waste: those activities, such as unnecessary documentation and long wait times that slow down the flow of value.

If we envision an ideal system, one that is 100% efficient, it is clear that we can increase scope and quality by using fewer resources and less time. Although physical constraints suggest we will never achieve perfection, it is still the target for which we must aim.

The Iron Triangle creates a mental block to progress.

It tells us we are doing our best: the relationships between outputs and inputs are fixed, and we can never expect to get any better. This is FALSE. It is time to jettison the Iron Triangle.

As product managers and product teams, we need to always look for ways to produce products more efficiently. Part of our job must be to spend time to reflect upon our current process, identify areas for improvement, and experiment with new ways of doing things. Some brave software developers did this in the early 90's and out of it came the Agile movement and substantial gains in quality, flexibility, and throughput.

It is product management's turn as a profession to question our beliefs and identify better ways to work.

The Importance of a Product Management Framework

What is a Product Management Framework?

In order to understand what a Product Management framework is, it's helpful to understand the literal definition of a framework: a basic supporting structure that underlies a system or process.

So a Product Management framework is what guides Product Managers and their company through every phase of the Product Management process and helps to keep a discipline that can be extremely complicated, as organized as possible.

Why is a Product Management Framework Important?

Team Unity

Not only is Product Management very complex, the details of how it's done often vary a great deal between companies and groups.

Having a proven Product Management framework which provides structure to everything that is done can be priceless for a Product Management organization. Since many Product Managers within the organization are coming from different backgrounds and only two percent of Product Managers have ever been formally trained in their job function, having a specific Product Management framework to work with keeps everyone on the same page.

Profitability

More and more, the importance of Product Management is being realized and appreciated (see our Critical Importance of Product Management video on Youtube.) Product Management is a key driver of both strategy and revenue in a company. In fact, in a recent survey which the 280 Group conducted of over 850 PMs and PM team leaders, on average respondents believed profits would increase by 34% if Product Management was fully optimized at their company.

One of the other key findings of the survey was that the companies that had a consistent process and framework in place had FAR fewer problems than the ones that didn't.

With Product Management so critical to the success of a company, it can be hugely profitable to improve it by integrating a comprehensive Product Management framework.

Comprehensive Work

Because the duties of a Product Manager are so extensive, it can be easy to let a few things slip through the cracks.

Managing everything that needs to be done is a serious challenge for PM organizations and that's why it is so important to have a framework in place. A good Product Management framework will take you through every phase of the product lifecycle and feature the most important tasks and key documents of each phase, from Conceive to Retire.

Following a Product Management framework will work wonders in ensuring that all tasks and documents are accounted for and no phase of the product lifecycle is neglected.

How to Choose a Product Management Framework

As you might expect, there are a few different Product Management frameworks that have been designed, so how do you decide on which model would be best for your organization to adopt?

Of course it depends somewhat on your specific situation but there are some important factors that should be considered.

Flexibility: Choose a framework that works with any development methodology, be it Agile or Waterfall, and isn't specific to one industry or type of product.
Covers the entire Product Lifecycle: Don't settle for a framework that leaves out or skimps on a particular aspect of the product lifecycle; all phases should be addressed.
Includes tasks and key documents: A Product Management framework will be the most helpful if it features both the specific tasks and key documents that go along with each step in the process.
Has training to go with it: Adopting a Product Management framework is a big step in the right direction for any PM organization but if the Product Managers go through training that is designed to get the most out of the framework, the benefits will immediately be taken to the next level.

Applying Systems Thinking to Your Product Development Process

Product Managers Are Outsiders in the Product Development Process

There are two underlying problems for Product Managers when considering any product development process, whether it is Agile, Waterfall, or a hybrid process.

The product development process is a subset of a larger product process (from product conception, to detailed business planning, all the way through end-of-life). However, while the product development process is usually well defined and defended, the overall product process is often ad-hoc, if it exists at all. Consequently, the more well-defined development process becomes the common understanding of the "product process," even though it lacks a complete view of how product value is created and measured.
Because the product development process is owned by engineering, the product manager has limited influence over the outcomes. As most product managers are well aware, there are hundreds, if not thousands of micro-decisions affecting the product that take place daily in the product development process. How can a product manager step in and lead when the process itself is owned by another part of the organization?
Agile methodologies attempt to address the second issue, and we can see this in scrum through the role of the product owner and the use of retrospectives.

But is there more that product managers can be doing to address these process issues? Yes, there is.

But as Einstein reminds us, the answers can't be found at the same level of thinking that created the problem. We need to take a step back and look more holistically at the system.

What is Systems Thinking?

In Peter Senge's highly influential book The Fifth Discipline, he describes systems thinking as "a way of thinking about, and a language for describing and understanding, the forces and interrelationships that shape the behavior of systems." Which sounds very conceptual and is hard to wrap our heads around. He goes on to explain,

"Have you ever seen in a family, people producing consequences in the family, how people act, how people feel, that aren't what anybody intends?' Yes. 'How does that happen?' Well… then people tell their stories and think about it. But that then grounds people in not the jargon of 'system' or 'systems thinking' but the reality – that we live in webs of interdependence."

Have you ever seen the product process, with its subset the product development process, produce unintended consequences? Produce defects, clash of egos/prioritizations, tradeoffs, compromises, or even end resulting products that were unintended?

Taking a high-level, interconnected system view of the product process can lead to new ways of understanding how products get defined, built, communicated, and maintained. And as a product manager, our role sits directly at the hub of the entire system, there is no one better placed or more directly responsible for understanding and guiding the system, than us.

The Iceberg Model

The Iceberg Model, formulated by M. Goodman in 2002, is a direct and practical way to utilize systems thinking. It states that the tip of the system iceberg looks at events, what is happening right now, and usually leads to knee-jerk reactions that are short-term and symptom-focused.

For example, let's say that engineering resources are moved from your project onto another project. You react vehemently defending your resources, going to your management and engineering management attempting to get resources back on your product. How successful are you?

Just below the surface of the water on the iceberg is a deeper level of thinking, and that is seeking out trends. Does this event happen often? What are the usual ways we solve this issue and does it have any long-term benefits or does it simply create more problems?

In our example, you could look at how often resources are moved between projects. Is there some way for you to predict or influence this event that you haven't seen or tried before? If it has happened before, what was the outcome to both your product, but also to the whole organization? Can you expect the same outcome this time? If you were to approach the problem with more trending data, would you be more successful in changing it?

Digging deeper into the iceberg, we find another level that explores the structures that create the trends. What is it about how our business or decision-making processes are set up that continue to create the same situation?

For our example, how is the overall value creation of products measured and is it even part of the decision making? What sort of risks is the company comfortable undertaking by shifting resources? What triggers the resource shift? Who is making the decisions and what data are they using?

Finally, we get to the deepest level of the iceberg, down in the murky waters that are the mental models of the organization. Here is where you evaluate what kinds of beliefs and assumptions are supporting the structures.

For our example, is there a fundamental expectation that PMs should be competing for resources? What beliefs exist around authority (both by the organization, and yourself) that would have the PM not be part of this decision-making? Do you feel powerless, or are you inserting yourself into the process?
Using Systems Thinking to Influence Your Product Development Process

As product managers there are some key ways to use systems thinking to influence all the processes surrounding our products, including the product development process.

Applying the ideas in the Iceberg model will help us move away from fire-fighting and start addressing root issues that impact long-term benefits. It will help align others through shared

thinking, and will reveal leverage points where we can make small process changes to get maximum benefits.

Iceberg Level 1 (Events): Get Past Your Own Reactivity

If we want to influence at a deeper level we have to stop being reactive, stop thinking at the most shallow levels, and bring ourselves into deeper understanding of the problems and what we're capable of.

Essentially, stop blaming others for our situations and see ourselves as powerful enough to make a difference in the system.

Iceberg Level 2 (Trends): Triangulate

Get new views of the trends that are taking place. One way of doing this is through feedback loops coupled with a mind to continuous improvement.

First, of course, take advantage of all engineering retrospectives that you can. But also implement feedback loops with other parts of the organization such as customer support, training, finance and legal, as well as with other product managers.

Find out what has been happening? Where are the changes? What can we predict? What are best practices and do they still work? Are there even better processes? Where are the breakdowns? This will give you insights into other people's processes, how to improve coordination with them, and move away from competition and into collaboration.

Iceberg Level 3 (Structure): Establish a Well-Defined Product Process and Set Product Goals

Two of the strongest ways to establish a structure to support your product are through a coherent product process and through the creation of clear product goals.

If you do not already have a well-defined product process, then get one. I am a big fan of The 280 Group's Optimal Product Process which I find to be thorough and flexible, but there are others. You need to learn then practice your product process; walk the talk, support it, get it sponsored, teach it to the organization and evangelize it. (The 280 Group has tools to help with all of this.)

Secondly, it is said that systems will self-organize around whatever goals are established, so you want to be driving the goals as much as possible. This means that you need to:

Establish clear product goals that people can feel motivated to achieve. ROI, roadmaps, MVP, personas, problem statements, user stories and acceptance criteria are all part of the product goals that impact the product development process. Make sure that you're including the opinions of all the relevant stakeholders when you establish those goals so that all parties will feel an ownership to achieve them.
Understand the competing goals. For example, all of your engineers all have personal goals set with their own managers that will determine if they are successful or not (if they will get a raise or not.) Do you know those goals? How can your product goals be framed to support those goals? Other products also have goals—how can you work with other product managers to

determine which set of goals across a portfolio will result in the highest value for the organization?
Iceberg Level 4 (Mental Models): Product Vision, Passion, Commitment

While it can be difficult to overcome some detrimental mental models, such as pitting product managers against each other to compete for resources, or sacrificing product value creation for a single individual's personal political gain, there are some tools in a product manager's toolkit here.

The most powerful tools are your product vision, coupled with your passion and commitment to that vision. We talked about how to create a product vision in a previous blog, so I won't repeat that information here. But hopefully now you can see why it is so important to have one, because it establishes the mental model for the product team.

And remember, there is not a single person in the organization that will ever have more passion or commitment to your vision than you, so it is important that you set the bar high, very high.

Give It Time

Finally, understand that it may take time to discern, develop, and apply alternative behaviors in your organization. People have been doing things a particular way, possibly for a long time, and find comfort in repeating old patterns. Keep learning, keep walking your talk, keep moving forward.

Elements of a Compelling Product Vision Statement

Do I Really Need a Product Vision Statement?

According to Les McKeown, CEO of Predicable Success, it is estimated that team members make between 20-200 micro-decisions a day that impact the success of a project. When you do the math on how many team members you have this leads to an astounding number of decisions that are being made daily around your product. A compelling product vision statement helps set the boundaries around how quickly and effectively those decisions can be made.

In addition, a compelling product vision statement inspires people to act on their own initiative without having to be pushed or pulled into action. Including a product vision statement as part of your strategic product planning will help move you from a product manager who is constantly running interference between confused stakeholders (i.e. “herding cats”) and into one who is driving an effective, collaborative, cross-functional team.

As Steve Jobs said, “If you are working on something exciting that you really care about, you don’t have to be pushed. The vision pulls you.”

Elements of a Compelling Product Vision

Let’s look at three key elements of a compelling product vision statement: Values, Vista, and Validity.

Values

A standard organizational vision statement is grounded in the values of the stakeholders it represents. These values trigger emotions that connect the stakeholders at a deep, powerful level to the vision.

A product vision takes that idea a step further, as it communicates at a visceral level the values of the customers. It should inspire among your stakeholders a high level of empathy for the buyers, users and influencers of the product. It is this empathy that becomes the inspirational force behind the stakeholders call to action.
As an example, according to the Inc. article “How Harley-Davidson Mastered Product Positioning”, this is the internal product vision statement for Harley-Davidson:

The only motorcycle manufacturer
That makes big, loud motorcycles
For macho guys (and “macho wannabes”)
Mostly in the United States
Who want to join a gang of cowboys
In an era of decreasing personal freedom.

You can see the difference between this internal statement and one of Harley-Davidson’s customer-facing tag lines “American by birth. Rebel by choice.” The internal statement gives a clear indication of the buyer and user persona but does so in terms the product teams can resonate with.

Vista

Vista represents the scope of your vision statement and it contains two components:

Time scale: how far into the future is your vision statement going to represent? Is it one product release (often used for waterfall development environments) or is it going to cover multiple or even on-going releases (used more often in Agile development environments). Whatever timing is appropriate for your release, make sure your statement links the work that is being done in the present to a future payoff for the customer.
Breadth of personas: who is the vision statement really about? Most internal product teams need to be connected to the primary buyer and user personas. However, more complex B-to-B-to-C products may need multiple or expanded product visions to help connect stakeholders to appropriate personas.
I found a Forbes article “Salesforce.com’s Marc Benioff On Innovation, Acquisitions And Reinvention” in which Marc Benioff demonstrates the difficulty, and the elegance, that I think can be found in more complex product vision statements.

He states:
“Somebody needs to be the platform that connects all these customers to these companies. I call it a one to one customer platform …That is the customer touch-point is prolific and changing and evolving. And you the company – Toyota, BofA, News Corp, whoever — you need to be able to have this consistent relationship with the customer wherever they/you are…”

Validity

A good product vision statement is grounded in reality. A couple of key things to keep in mind when determining the validity of your vision statement are:

Timing: is this the right time for this group of stakeholders to take on this vision? For example, if your team has spent several cycles in chaos and are finally coming together, this would be a tough time to sell a vision that promotes radical change. If on the other hand the stakeholders are hungry for change, don’t bring them a status quo vision.
Credibility: does this group of stakeholders have what it takes in terms of tools, training, financing and core competencies to pull off the vision? Do you as the leader have a track record of success for this type of vision? If not, people may feel set up to fail.
Organizational Alignment: does your vision statement fit within the overall mission and vision of the organization? If not, who is sponsoring this product vision?
Positioning Statements as Product Vision Statements

I have often found the easiest way to construct a compelling product vision statement is by starting with the product positioning statement. A position statement goes like this:

For (target customer) who (statement of the need or opportunity) the (product name) is a (product category) that (statement of key benefit – that is, compelling reason to buy)
Unlike (primary competitive alternative) our product (statement of primary differentiation)
As an example, here is a former product vision statement for the Blackberry:

For business e-mail users who want to better manage the increasing number of messages they receive when out of the office, BlackBerry is a mobile e-mail solution that provides a real-time link to their desktop e-mail for sending, reading and responding to important messages. Unlike other mobile e-mail solutions, BlackBerry is wearable, secure, and always connected.

What I found interesting in this statement is that they are not targeting mobile users, such as a sales team member, but that the solution was mobile.

In summary, a critical tool for product leadership is the ability to create a product vision statement that will engage passions, inspire action, and guide decision-making.

No matter how you do it, in a few words, a long statement, or through your positioning statement, the product vision statement, done well, can unify and motivate your product team.

Product Owner vs. Product Manager Exploration

Regardless of which role or title you may have, you should have at least a basic understanding of the Product Owner vs. Product Manager comparison; similarities, differences, and overlap.

Why? In our recent Product Management Team Survey we found that in companies doing Agile development, 70% indicated that the Product Manager is also the Product Owner.

If you don't have a good understanding of both roles (whether you are doing both or just interacting with someone doing one of them) you won't be nearly as effective in leading your team.

This article discusses some of what we teach in our Agile Excellence for Product Manager and Product owners training course that includes the worldwide-standard certification credential.

Product Owner vs. Product Manager Background

As you have almost certainly noticed by now, Agile development methodologies have taken the product world by storm and don't seem to be leaving anytime soon. Our study indicated that 69% of teams are doing some form of Agile.

We at the 280 Group have specialized in helping Product Managers learn how to be great for over a decade, and now with Agile development so entrenched in much of the product industry and intertwined with Product Management, there is a new role that has become important to understand: the Product Owner.

The scrum framework has included the product owner as one of its defined roles in Agile and so naturally this has spawned a great deal of Product Owner vs. Product Manager confusion and debate.

Before we go any further, it should be noted that this is a discussion about Product Owner vs. Product Manager as roles as opposed to titles. For example, you may have the title of Product Manager and be acting in the role of Product Owner.

The Product Manager Role

Product Management is a broad discipline which comes with a vast range of activities and responsibilities. In general, the Product Manager is responsible for driving the strategy and overall success of the product.

The job of Product Manager may be specifically defined differently across different businesses but if you engage in any of the following responsibilities you are acting as a Product Manager.

Business/Strategy Focus

As a Product Manager you should be the market expert because you have listened deeply to you customers. Product Managers are the authority on anything regarding the landscape of the market and act as the voice of the customer in all situations.

You own the product strategy which drives the vision and roadmap, and manage the product throughout its entire lifecyle, not just the development phase.

Product Managers must focus on the notion of the problem space and avoid specifically defining the product. They present needs in the form of user stories (when doing Agile) and market requirements (when doing waterfall/phase-gate development) and let the engineers create the solution to solve the problems.

You must ensure that there is a business justification for your decisions, and make the tough trade-offs with regard to features, schedule, and costs.

Ultimately Product Managers deliver differentiated products to market based on a winning strategy the have defined, and their role traditionally belongs to the “business” side of things.

The Product Owner Role

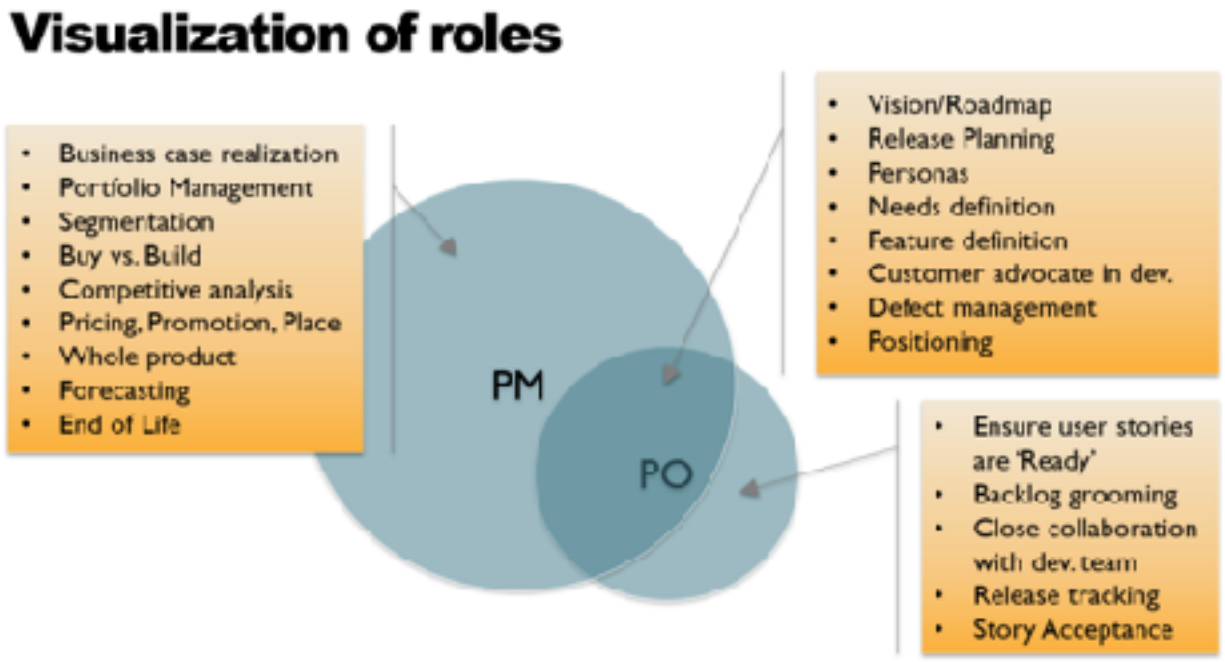
Much like the Product Manager role, Product Owners’ tasks may vary depending on the situation but based on the Scrum Guide, there are a few special tasks that absolutely need to be done by the Product Owner.

Features/Development Focus

It is the job of the Product Owner to maximize the value of the product and the work being done by the development team.

As a Product Owner you are the sole person responsible for managing and ordering the product backlog.

Product Owners prioritize and manage user stories and need to work closely with the development team to ensure they understand what each item in the product backlog is and why it is important.



Product Owner vs. Product Manager Role Visualization

The Big Issue is That All These Tasks Need to Get Done

You need to have clarity on who has responsibility for each of these items.

When looking at the Product Owner vs. Product Manager graphic above you can see that Product Owners are responsible for more of the detailed work and how closely they work with the development team.

There are many tasks that you can find in the top right corner of the graphic which overlap between the PM and PO roles; when allocating these, it may be helpful to take individual strengths and weaknesses in to consideration.

In the top left portion of the chart, you see a set of tasks that are outside the scope of what Agile defines as that of the Product Owner.

If you only remember one thing from this article, it should be how things are broken out in the graphic. Make sure that it is clear who does what and that nothing is left out.

This is one of the biggest mistakes that Agile teams make – they have someone who focuses on the Product Ownership duties and then the strategic, vision and other activities aren't owned by anyone and don't get done.

This means that the strategic focus is greatly diminished or lost. The result is that the product may meet needs short-term for customers but ultimately fails in the longer-term.

Many organizations struggle with dividing their Product Owner vs. Product Manager responsibilities and if someone has all of these tasks assigned to them, the workload becomes overwhelming, and at this point the focus becomes what is urgent and detailed. One way to combat this is to make sure that both the PM and PO have gone through training and obtained a certification so they can be more productive.

The Product Owner vs. Product Manager comparison is one of similar but distinct roles and while the division of roles may vary, it is critical for the product and the business as a whole that all the responsibilities are adhered to.

Product Manager Skills: Maximizing Influence Through People Skills

Product Manager Skills: Isn't Being the Expert Enough?

You're the expert in your product, your market, your customers and your competitors. Yet, despite your expertise you find yourself struggling with internal stakeholders about schedules, feature prioritization, design, pricing, messaging, or product strategy. While some of this struggle is natural – you work with smart, opinionated people who hold a piece of the puzzle, it is also incredibly frustrating. What was the point of all your product manager skills and effort to become the expert if people still argue with you about your conclusions?

Maximizing Influence Through People Skills

My first Leadership Instructor taught me that Credibility = Expertise + Trust.

Your expertise is only half of the product management skill set needed for credibility—you also need to build and maintain trust. While we want to believe that logic influences people, the reality is that we are emotional beings, and logic will only go so far.

Your stakeholders are people who make decisions from their gut emotions, whether they admit to it or not. Thus, trust becomes essential to Product Management influence.

With trust comes:

The ability to communicate more clearly.
People trust they are being heard and are more likely to listen more closely to you. They believe your plans incorporate success for them as well as for yourself, and consequently do not need to fight as hard for their view of success.
An increase in innovation.
Trust allows people to fail and recover, a critical component of innovation. As a Product Manager you are in a position to encourage innovative thinking, offering support to new ideas and protecting people when new ideas fall flat.
An increase in stakeholder ownership and alignment.
Your stakeholders will invest in someone's vision of the product, either yours, their own, or someone else's. If they trust you on a gut level, they are far more likely to align themselves to your vision.
People Skills That Build Trust

Our goal in exercising "people skills" is not to make other people feel good, but to maximize the amount of trust we are building and maintaining. Below are a few examples key product manager skills that build trust:

Integrity—Walk Your Talk
Keep your word. Admit and correct your mistakes. Own your power and passion. It is important that your stakeholders learn who you are and can count on that consistently.
Listen First, Speak Last
Real listening is 90% of solving disagreements. You want someone on your side? Listen to them first. They'll feel validated having been heard and you'll know best how to frame your opinion. And you may learn something new, too!

State solid YESes and NOs

Every Yes to one thing is a No to something else, and vice versa. Clearly delivering a Yes or No answer provides focus on what is truly important to a successful product. The ability to withstand the discomfort of the moment of telling someone something they may not want to hear informs your stakeholders that you are clear on what needs doing and you're willing to face hardships to get there.

Know Other People's Jobs

Know what your stakeholders do and how they contribute to the success of the product. Know their processes. Know how they succeed at their job and how you can help them. Not only does it help create empathy for the challenges your stakeholders face, but when seemingly unsurmountable problems arise (and they will) you'll know where to push, pull, or side-step the system, and who can help you.

Get to the Point Quickly

With all our market and product knowledge we can sometimes try to communicate too much. If you listen to how leaders communicate you'll realize they get to the point without all the backstory. You can do the same—be polite yes, but also state your need without the backstory unless asked for it. You'll be amazed at how this skill can transform your collaborative relationships!

Take All of the Blame and Give Away All the Credit

Whenever possible, take the blame for problems. This sounds counter-intuitive at first, but it works. I'm not suggesting you take the blame for illegal or unethical actions taken by someone else, but when it comes to your product the person who holds the blame is the one who made the decision, and this subtle distinction contributes to the Implied Authority that you are the decision-maker for the product.

From your executive's perspective, they generally don't care who is to blame, they want to know that someone is aware of and correcting the problem and won't let it happen again. They want a responsible leader and you want to visibly be the person in this role.

From the product team's perspective, they want to know you'll shield them from blame. You provide a safe path to success for them.

Of course, your next step is to follow through privately with the person who is responsible and make sure the problem is cleaned up, that gets back to reinforcing your integrity. And finally, make sure to give credit for all product success to your team members, not take it yourself. This practice of publically falling on your sword while simultaneously giving away credit will do more to build trust, to establish your authority and leadership, than anything else.

Adding these people skills to your collection of product manager skills will increase your ability to drive a successful product. These are key skills in the path from Product Manager to Product Leader.

Be stubborn on vision but flexible on details.

The essence of strategy is choosing what not to do.

When the wind blows, some people build walls, others build windmills.

Make every detail perfect and limit the number details to perfect.

The value is in what gets used, not in what gets built

Simplicity is not the absence of clutter, that's a consequence of simplicity. Simplicity is somehow essentially describing the purpose and place of an object and product. The absence of clutter is just a clutter-free product

Only move forward with creating a product that will be "above the bar."

Any damn fool can make something complex, it takes a genius to make something simple.

If you are not embarrassed by the first version of your product, you've launched too late.

The key is not to prioritize what's on your schedule, but to schedule your priorities

The ability to simplify means to eliminate the unnecessary so that the necessary may speak

The role of leadership is to transform the complex situation into small pieces and prioritize them

Don't make a better [x], make a better [user of x]

We see our customers as invited guests to a party, and we are the hosts. It's our job every day to make every important aspect of the customer experience a little bit better.

Our job is to invent on behalf of our users

our most unhappy customers are your greatest source of learning

Data beats opinions

The only way to win is to learn faster than anyone else

My biggest regrets are the moments that I let a lack of data override my intuition on what's best for our customers.

Feedback is the breakfast of champions

“When we create stuff, we do it because we listen to the customer, get their inputs and also throw in what we’d like to see, too. We cook up new products. You never really know if people will love them as much as you do.”
– Steve Jobs

“If you are not embarrassed by the first version of your product, you’ve launched too late.”
– Reid Hoffman, LinkedIn

“Make every detail perfect and limit the number details to perfect.”
– Jack Dorsey, Square

“If you keep your eye on the profit, you’re going to skimp on the product. But if you focus on making really great products, then the profits will follow.”
– Steve Jobs

“If you continue to improve a product enough, you’ll eventually ruin it.”
– David Pogue, the New York Times

“Management is doing things right; leadership is doing the right things.”
– Peter F. Drucker

“Be stubborn on vision but flexible on details”
– Jeff Bezos

“Take ice. Ice is fascinating to me. Ice is the one thing in our world that went from an agricultural product to being manufactured.”
– Alton Brown

“The details are details. They make the product. The connections, the connections, the connections. It will in the end be these details that give the product its life.”
– Charles Eames

“Roadmaps are evidence of strategy. Not a list of features.”
– Steve Johnson

“Why do so many professionals say they are project managing, when what they are actually doing is fire fighting?”
— Colin Bentley

“Trying to manage a project without project management is like trying to play a football game without a game plan.”
— K. Tate

“Of all the things I’ve done, the most vital is coordinating the talents of those who work for us and pointing them towards a certain goal.”
— Walt Disney

“If it wasn’t for the ‘last minute’, nothing would get done.”

“Warning: dates in the calendar are closer than you think.”

“A project without a critical path is like a ship without a rudder.”
— D. Meyer, Illinois Construction Law

“Project management is like juggling three balls – time, cost and quality. Program management is like a troupe of circus performers standing in a circle, each juggling-three balls and swapping balls from time to time.”
— G. Reiss

“A good plan can help with risk analyses but it will never guarantee the smooth running of the project.”
— Bentley and Borman

“PMs are the most creative pros in the world; we have to figure out everything that could go wrong, before it does.”
— Fredrik Haren

“If, on your team, everyone’s input is not encouraged, valued, and welcome, why call it a team?”
— Woody Williams

“After the launch phase, your product is old news. Take advantage of the opportunity to generate interest when your product is new.”

“Embrace iteration as the road to improvement, but don't let that lull you into rolling out poorly-thought-out crap.”

“One reason product management is such an appealing career is you get to sit at the intersection of technology, business, and design.”

“No offense but a seller will say whatever it takes to sell a product but if you sum it up, it comes down to hard work.”

“Engineering-driven companies falsely assume that because they build it, the industry will magically become aware and be willing to buy it.”

“Not having a specified launch plan and process is one of the biggest pitfalls in the technology market.”

“Methodology must be flexible. Companies often don't adopt the materials & methods they were trained on because they aren't flexible enough.”

“Doing testing with real users (in addition to internal quality testing) can help you avoid the serious embarrassment of a failed product.”

“Only move forward with creating a product that will be "above the bar.”

“Don't focus all your time and effort on creating the templates & perfecting the documents. Answering key product questions is more critical.”

“In our factories, we create flexibility by paying more to workers who can work at more stations on a production line. We value flexibility, and we pay for it. In contrast, most product development organizations exclusively reward specialization.”

Product Managers need to be people of action. While it's good to have a good discussion and some data on solving specific problems, make sure taking action is in the forefront of your life. How much discussion you have and how much data you collect will probably reflect the size of your company. Startups can't afford to invest in long drawn out studies and need to move quickly and large companies typically spend more time researching, planning, and collecting data. Always remember your product solves a customer problem, be actively moving in that direction.

Action is the foundational key to all success.

– Pablo Picasso

Don't wait. The time will never be just right.

– Napoleon Hill

Great thoughts speak only to the thoughtful mind, but great actions speak to all mankind.

– Theodore Roosevelt

Be the change that you wish to see in the world.

– Mahatma Gandhi

Product Managers need to have a good attitude. People are drawn to others with a good attitude. As a Product Manager, you will need to influence customers, product development, marketing, and your management. Having a good attitude makes a difference. Don't get me wrong, I don't believe attitude is everything. Attitude doesn't replace competence or leadership, but it can be the difference maker in bringing the team together to rally behind the product. Always remember to have a good attitude.

Attitude is a little thing that makes a big difference.

-Winston Churchill

Keep your thoughts positive because your thoughts become your words. Keep your words positive because your words become your behavior. Keep your behavior positive because your behavior becomes your habits. Keep your habits positive because your habits become your values. Keep your values positive because your values become your destiny."

– Mahatma Gandhi

Ability is what you're capable of doing. Motivation determines what you do. Attitude determines how well you do it.

– Lou Holtz

It is your attitude, not your aptitude, that determines your altitude.

– Zig Ziglar

Product Managers need to be persistent. As the leader of your product, you need to persistently press on toward success. Always remember that because you have thought through a problem doesn't mean your team has had the opportunity to think it through. Be persistent in your communication. Remember that customers are dealing with the product today, with all it's flaws and not with the vision of where you plan to take the product. Be persistent in overcoming today's problems and listening to the customer. Just because your the Product Manager doesn't mean everyone is going to agree with you. Be persistent in connecting with people. In everything you do, be persistent.

Success consists of going from failure to failure without loss of enthusiasm.

– Winston Churchill

The man who removes a mountain begins by carrying away small stones.

– Chinese Proverb

Edison failed 10,000 times before he made the electric light. Do not be discouraged if you fail a few times.

-Napoleon Hill

I've failed over and over and over again in my life and that is why I succeed.

-Michael Jordan

Product Managers must have confidence. Have you ever been around a leader that doesn't have confidence? Who would ever follow someone who isn't confident in where they are going? As a product manager, everyone will be watching you very closely. Your teams confidence in the product will reflect your confidence in the product. It doesn't mean you are not realistic in where the product is today, your confidence must show that you are confident in the team to solve the problems that lay ahead of you. In everything you do, be confident.

Experience tells you what to do. Confidence allows you to do it.

– Stan Smith

Go forward confidently, energetically attacking problems, expecting favorable outcomes.

– Norman Vincent Peale

Product Managers must be leaders. It's easy to be a leader when the people you lead report to you. It takes real leadership to lead in today's matrix managed teams. In order for a Product Manager to be successful, you will need to lead product developers, marketing managers, sales teams, and customers, in which none of them will report to you. As a Product Manager, I became a student of leadership. One of my favorite authors is John Maxwell and one of my favorite books is the A 360 degree leader:Developing Your Influence from Anywhere in the Organization.

Leadership

Life is like a coin. You can spend it any way you wish, but you only spend it once.

-Lillian Dickson

Management is doing things right; leadership is doing the right things.

-Peter F. Drucker

Last and probably most important is that Product Managers must be life-long learners. As a Product Managers you need to become good at learning and learn something new everyday. Whether it's a new product feature, a new technology, a new process, a new application, a new competitor, or a new way of thinking, learning is vital to the life and success of a Product Manager. There are new technologies emerging everyday that can drastically effect your product. Product Managers need to be learning everyday to face the difficult landscape that will throw problems at you from every direction. Learning from everything you do and from the leaders in your industry will help you to become a better Product Manager.

The only thing that interferes with my learning is my education.

– Albert Einstein

Try not to become a man of success but a man of value.

– Albert Einstein

Tell me and I forget. Teach me and I remember. Involve me and I learn.

– Benjamin Franklin

As a Product Manager words are a powerful thing and will have a huge impact on your team. Start focusing on the words you use today and make it a goal to improve your communication skills.

CI/CD automation (Part I)

“Continuous Integration”
What is CI?

Continuous integration (CI) is a DevOps software development practice where developers regularly merge their changes into a central repository, for automated integration and test case executions. CI is a development practice that requires developers to integrate code into a repository several times a day. By integrating regularly, developers can detect errors quickly, and locate them more easily.

CI saves cost of development. Lack of CI usage increases the time between two integrations which exponentially increases the difficulties in finding and fixing the problems. Such integration problems can be easily knock-off a project off its schedule, or may lead to a project failure.

CI Fundamentals

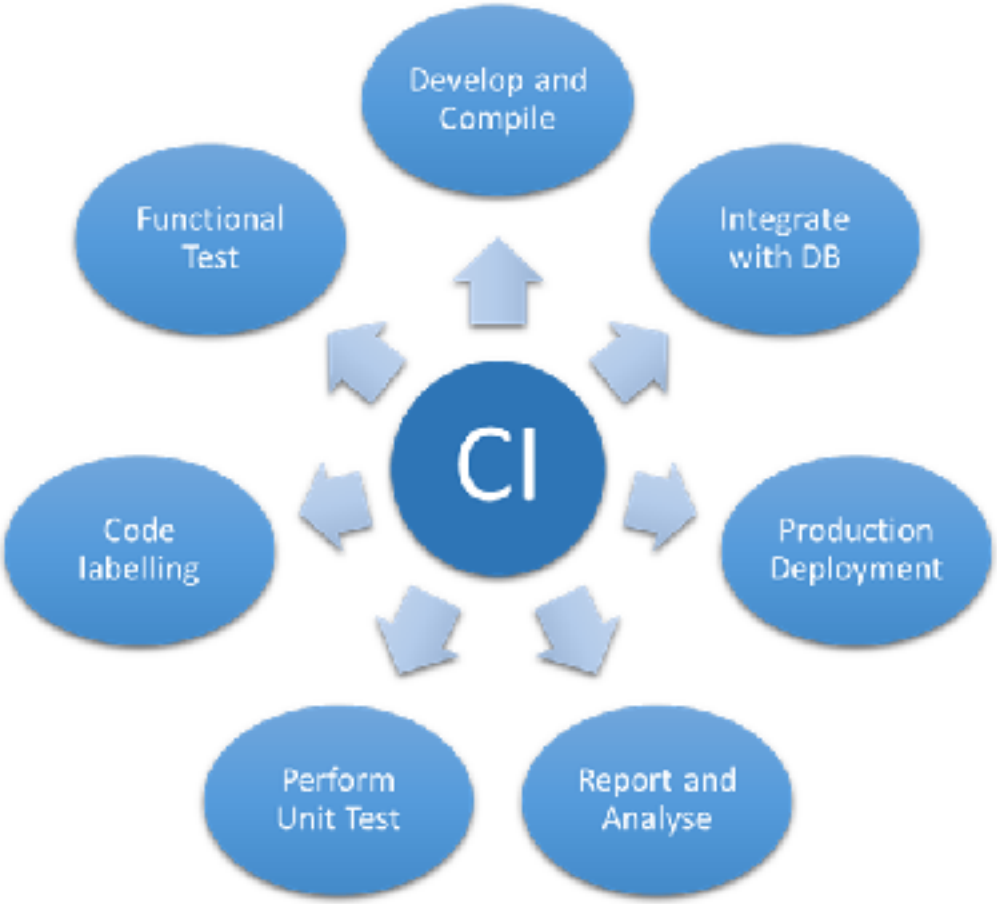
A key principle of a Continuous Delivery approach is that the application is always production ready. This implies that all artifacts required to build the application are stored and managed in a version control system, and that code must be regularly merged with the source mainline to ensure small increments between builds.

There are three important and fundamental elements that underpin any Continuous Integration System.

1. Commitment to building tests: The commitment of the development team to produce a comprehensive test suite at the unit level and functional level together with their code is the most important element of a CI system. This element is essential to achieve the “guaranteed” level of code quality that can be production ready at any moment.

2. Never break the build: The goal is that the application must be ready to be built, packaged and deployed on every committed change.

3. Version Control: Another fundamental element to Continuous Integration is version control. To be ready to focus on implementing Continuous Integration, the code needs to be managed by strict version control policies



Why CI?

There are many languages developer can choose from, different test suites, countless frameworks and many CI offerings. CI is a way to increase code quality without putting an extra burden on the developers. Tests and code checks are handled on a CI server. In case of any error, it is automatically reported back to developers.

Here are some of the reasons why any organization should implement CI in their project,

1. Allow developers to run their tests in the real world:

When a developer writes a code, the tests are successfully passed on his/her machine but the same fails on someone else's. CI can save such ambiguities. Developer just have to push their code to the new branch and CI server will take care of running test

2. Improve Developer productivity:

Continuous integration helps teams to be more productive by freeing developers from manual tasks that help to reduce the number of errors and bugs released to customers.

3. Find and Address Bugs Quicker:

With more frequent testing, Team can discover and address bugs earlier before they grow into larger problems later.

4. Deliver Updates Faster:

CI helps team to deliver updates to their customers faster and more frequently

5. Increases trust in the software quality:

By continually building the software, increases confidence that the software is of high quality

CI Goals

CI is most essential element during the build or integration stage of the software release.

The key goals of continuous integration are

To find and address errands quicker
Improve software quality
Reduce the validation and release time for new software updates
“Continuous Delivery”
What is CD?

Continuous delivery (CD) is a DevOps software development practice where the integrated changes are automatically built, tested, and prepared for a release to production. CD is the ability to get changes of all types including new features, configuration changes, bug fixes and experiments into production, or into the hands of users, safely and quickly in a sustainable way. When continuous delivery is implemented correctly, developers will always have a deployment-ready build artifact that has passed through a standardized test process.

Principles of CD

1. Automate Everything

A manual deployment can never be described as repeatable and reliable. Company has to invest seriously in automating all the tasks that they do repeatedly, and this tends to lead to reliability.

2. Everybody has responsibility for the release process

Developers should develop projects with a mind for how to deploy them. Project managers should plan projects with attention to deployment. Testers should test for deployment defects with as much care and attention as they do for code defects.

3. Improve continuously

Continuous improvement means system will always be evolving and therefore easier to change when needs be.

Continuous Delivery

Why CD?

1.Low risk releases:

The primary goal of continuous delivery is to make software deployments painless, low-risk events that can be performed at any time, on demand.

2. Higher quality:

When developers have automated tools that discover regressions within minutes, teams are freed to focus their effort on user research and higher level testing activities such as exploratory testing, usability testing, and performance and security testing.

3. Lower costs:

Any successful software product or service will evolve significantly over the course of its lifetime. By investing in build, test, deployment and environment automation, an organization can substantially reduce the cost of making and delivering incremental changes to the software by eliminating many of the fixed costs associated with the release process.

4. Better products.

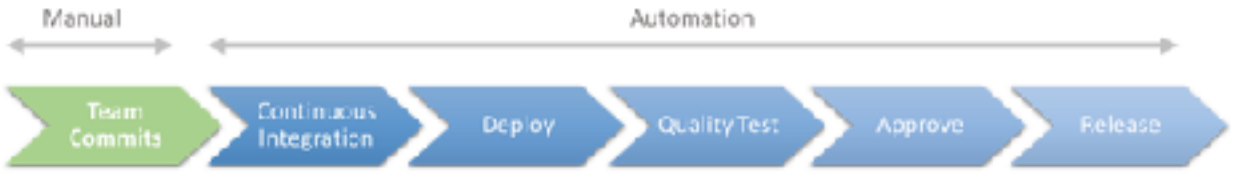
Continuous delivery makes it economic to work in small batches.

Goals of CD

Automate the Software Release Process
Improve Developer Productivity
Deliver Updates Faster
“The CI/CD Pipeline”
CD/CI pipeline is an important aspect of a software project. It saves a lot of manual and error-prone deployment work. It results in higher quality software after continuous integration, automated tests, and code metrics.

With a CI/CD pipeline, for every change in software code, it is built and tested automatically. Code analysis are executed against it. If it passes quality control gates along with the required tests, it is automatically deployed. The automated acceptance tests are executed against it. This process of quality control and automation is getting more and more necessary in today’s fast-paced software-centric environment where companies have to release stable versions of software frequently.

CI/CD pipeline consists following steps:



The Continuous Delivery Process

Commit: When a developer finishes a change to an application, he or she commits it to a central source code repository
Build: The change is checked out from the repository and the software is built so that it can be run by a computer.

Automated tests: The change is tested from multiple angles to ensure it works and that it doesn't break anything in the system/software
Deploy: The built version is deployed to production

Benefits of the CI/CD Pipeline

1. Developers have greater control over the process and can respond more quickly to market demands by quickly bringing up new environments, testing against them and easily starting over, if required. Moreover, developers can quickly adjust to new environments — an approach that has proven to decrease failures and resolution time when they do occur

2. Operations benefits as reduced friction in the CI/CD pipeline created by automation decreases repetitive and manual work along with the opportunity for "fat finger" mistakes thus improving efficiency of the process

3. Less manual and repetitive work provides everyone in the process with more time for strategic work, which provides direct bottom-line value to the organization

Conclusion

The greatest benefit of CI/CD automation is reduced risk. Many a times the issues with delayed and infrequent integration manifest during the on-going process of project development, where the stakes are high and pressure to deliver is greatest. Further, it provides easy way for automation at many levels, starting with build automation, test automation and more recently continuous delivery, all of which aim to encourage the creation of higher quality software, faster.

Product management is a coveted role.

Ask aspirants, and you'll get answers like 'I want to take strategic decisions', 'I want to bring my vision to the market' or even 'I realized that the product I am developing for does not satisfy consumer demands. My manager does not see it this way, and just wants to finish the project. I feel I can do a better job at defining a product, so I want to try my hand'

The definition of a product has morphed significantly with the evolution of new businesses around technology and software. As products get more complex, the sheen around product management has intensified.

Larger companies usually have managers specializing in product and business strategy, product development, product marketing, go to market and analytics; but these boundaries are often nebulous. As a product manager, one might end up doing some or all of these, and often looking at other aspects like marketing, vendor management, business development, etc. It's easily one of the most complex roles in an organization.

The problem is: if you're entering it fresh, there's not much in the way of preparation that you can access. There's a lot of hands-on learning, and the early years as a product manager can be tough.

I made a lot of mistakes early on in my career as a PM, but many taught me important lessons. Here are five lessons I've learnt along the way.

1: The importance of the why-what-how cycle

When you hold a product in your hand, you see the end-result of a lot of work. Product management deals with the journey from the 'intent' of building the product to its final expression as a product or utility.

This journey starts with the 'why', moves to 'what' and then through the 'how' phases.

The 'why' phase includes understanding why you're doing what you do – consumer needs, softer aspects like behavior, habits, etc., why consumers would pick your product over competition, etc. This is the least deterministic of all phases. There are usually more questions than answers, and the answers themselves are usually best estimates based on info you have at hand.

Good product managers are able to absorb a lot of complexity at this phase, and start bringing structure for the next phase.

The next phase is 'what' do you build, 'what' features you put into the product, 'what' user experience works best, etc.

The last phase is the ‘how’ do you do it. This involves executing the project with your design and development teams, iterating, etc.
When I started working as a PM, I rushed into the 'how' part without really thinking of the 'what' and the 'why'. It was only later that I realized that getting the 'why' clear meant a lot of effort saved downstream.

2: Appreciating the effort and factors to make a product successful

A lot of product management is about communicating across functions.
For a product to be successful in the market, a lot of items have to fall in place: product experience has to be great, marketing should identify key messages and create collateral around it, vendors have to be invoiced and ready, pricing has to be right, sales guys have to be trained, distribution has to be sorted, customer support and service teams have to be primed….the list goes on.
Many of these functions kick in at different points of the product development process. As a novice, one often does not realize what goes into creating a successful product, and focuses only on a couple of areas.
When I started, I didn't realize that the product development process is just 30% of the job done. A whole lot of work needed to be done *after* the development process.
What I learnt: Pick friends in finance, procurement, marketing and online marketplace/retail and try and understand their roles and pain points. It helps round off the experience, and prepare for the role after building the product.

3: Not trying to satisfy everyone

Even after you have built your product roadmap, you will receive numerous requests for feature modifications from various functions. Marketing may ask for enhance a key selling point so that they can pitch better. Sales may ask for a lower pricing variant for a different consumer segment. Your development team may realize that plans laid earlier need changes. There will be priority bug fixes to handle along with new feature requests.
The reality is: time and resources are finite. Feature requests aren’t.
A PM needs to know when to say no to requests, and what to say no to.
In the early years as a PM, this was often difficult: almost everyone around in the project I worked on had been on it longer. I had reached out earlier for advice to many of them, and often did not have the confidence to back up my viewpoint when it differed from theirs.

The downside risk is probably the highest for this. Most products falter during development because the requirements keep changing or new requirements are added. Product shipment is delayed. This may result in being late to the market, or significant revenue losses.

I was tempted to try building a consensus opinion on decisions. While that works in some situations, it also led to slow and leaderless decision making, affecting the end product. It was something I had to give up to be effective.

What I learnt: Review and keep in mind the top three features of your product for your target segment. Spend time on refining these, instead of adding new features that offer only incremental benefit. Identify minimum threshold requirements for the other requirements and keep the other requirements optional.

4: Not getting emotionally attached to the idea/product

Most popular literature advises that passion is a necessary ingredient when you're building products.

That's certainly true. Before the glory of a successful product launch, you face innumerable days of planning for the future, and staying committed to the end result all through. That's not possible without having a passion for the product.

But there's a difference between being passionate and being emotional. This is seldom easy to decipher.

Being passionate means that you're planning for the success of your product, but are flexible enough to incorporate other viewpoints. Passion is inclusive and inspires others to give their best.

In contrast, being emotional results in trying to control the process too much and attempting to get people to always accept your point of view. You risk repulsing other ideas, and people washing their hands off – let him/her handle the mess, I tried to advice but s/he did not take it.

If you're extremely emotional, you believe that product *cannot* fail, discounting possibilities that things can go wrong. As things go wrong (and they will!), you expend energy agonizing about why things are not working out, instead of facing the reality, making new decisions and moving on.

It's a difficult balance to achieve, and one that did not come easily. Only with experience did I realize that I had to let go of the 'ownership' angle to a product to really listen and accept feedback, especially that which was not encouraging.

5: Don't start a post-mortem immediately when things go wrong

During any product development process, you will encounter items that do not go according to plan. Some of that might be due to a mistake that you are accountable for.

How do you react?

As a PM, you're often steering a ship, even if that's just struck an iceberg. Panic can ruin everyone's confidence and result in ruinous decisions.

Most experienced PMs have a mental plan of what could go wrong in situations, and have thought through what they could do. There are extreme cases that you may not be able to avoid, but if you have though of eventualities other than success, you can start putting an alternate plan together to get out of the mess.

The worst, but easiest mistake to make at this time: starting a post-mortem of why things went wrong and who was responsible.

One of the lessons experience taught me is to avoid getting into this mode when crisis struck, but to resolve the crisis first and do the post-mortem later.

USER STORY REFLECTIONS

User stories are a simple, straightforward tool. But successfully applying them can be surprisingly hard. This post offers four refections to help you improve your user story practice.

Users
As its name suggests, a user story describes how a user or customer uses the product–a digital product is captured from the perspective of the users. This avoids a solution-centric view where we worry more about how to provide and implement the product features than why and how people will use them.

Understanding who the users are and how the product will benefit them is central to discovering and creating the right stories. In other words, if you do not know who your users are and what problem they want to see addressed or which benefit they would like to experience, then you should not create stories! Otherwise, you are in danger of speculating about the product features rather than deriving them from the user needs.

I'd like to take this further and suggest that you should meet the users. This is not to say that the users will be able to correctly tell you what the product should do—it's your job to figure this out. I know that product managers and product owners don't always have access to the users and are sometimes shielded from them by the sales team or management. I also understand that development team members tend to see users even less frequently. But if you want to leverage user stories to develop a successful product, then you should do try to observe and talk to the people who will use it. How else can you truly understand their needs and empathise with them?

Conversation
When I first came across user stories over 15 years ago, I was surprised by their lack of detail. I was used to working with use cases, and it took me a while to get my head around their sketchiness. They seemed less like requirements and more like notes. And that's exactly what stories are meant to be—notes that capture the essence of a conversation.

On its own, a user story is pretty useless. It needs to be complemented with a conversation between the person in charge of the product, the product manager or product owner, and a cross-functional development team.

Maybe it's helpful to take a quick look at the context in which user stories emerged. Stories were first used in Extreme Programming in combination with the role of an onsite customer. The onsite customer is a member of the user community who is collocated with the development team. The individual would discuss with the team what should be built in the next iteration, and captured the conversation as stories to remind everyone of what was discussed.

Agile approaches happily accept tacit knowledge, knowledge that is not explicitly expressed and written down, and they prefer face-to-face conversation over documentation. Why? To speed things up, to allow the team to quickly implement some functionality and to validate it by exposing it to the users.

I find it even better to co-create user stories by inviting the development team members to help discover new stories and amendments to existing ones—based on the feedback and data

gathered from the users. This leverages the team's collective knowledge and creativity and tends to result in better, clearer stories. To do so, make collaborative story work part of your product backlog grooming or refinement process.

But if you cannot have at least a meaningful conversation, if product owner and team cannot discuss new stories together so that the team understands the stories and can make suggestions for improving or splitting them, then you should not use user stories. The conversation is not optional; it is an essential part of user stories. Without it, the story is not usable or it becomes bloated with details. If you find that you need to capture more details or hand-off requirements, then use a different approach like use cases instead.

Use cases tend to be more effort to create and update but they offer more structure. They come with pre- and post-conditions, triggers, a main success scenario, alternative success scenarios, and exception scenarios. This allows you to describe the product functionality in more detail, which may be necessary when working with remote teams.

Simplicity

User stories are a very simple tool—we simply tell stories about how users are likely to interact with the product and capture their essence. That's it. Unfortunately, I have seen a fair amount of stories that were far from simple and straightforward.

I find it helpful to remember that in the early days of agile, people didn't talk so much about user stories. Instead, they used the term story cards. The reason for this is simple: stories were captured on paper cards. These have two advantages: they facilitate collaboration and visualisation. Everybody can write on a piece of paper, and the cards can be easily arranged on the table or office wall. Electronic tools are less conducive to teamwork in my experience; one person is usually in control of the keyboard and tool. What's more, it's much harder to display the stories and prevent them from being hidden away. I would therefore encourage you to experiment with paper cards to capture your stories—at least when you create new stories. You can key them into your favourite electronic tool afterwards if you wish. But be warned: I've seen teams who decided after some experimentation to switch from electronic to paper-based stories.

Another area that can benefit from simplification are the acceptance criteria. As their name says, the criteria want to communicate what needs to be in place so that the story can be considered as done and the corresponding functionality can be exposed to a user or customer. Acceptance criteria should be no big deal but follow naturally from the narrative by asking, "how can we tell the story is done?" If you make the criteria more complicated, the stories become harder to understand and start to slow you down.

Limits

User stories are great at capturing product functionality, things people can do with a digital product like searching, evaluating, and purchasing products online. You can also use stories to capture non-functional aspects of a product, such as performance, robustness, and interoperability.

But when it comes to the user interface and user interaction, user stories are less suitable. Sketches and mock-ups are better at describing the UI design. And scenarios, workflow diagrams, storyboards, and story maps are better suited to capture user interactions that comprise several steps.

Additionally, writing user stories is worthwhile when you develop software that's likely to be reused. But if you want to quickly create a throwaway prototype or mock-up to validate an idea, then writing stories may not be necessary. Remember: User stories are not about documenting requirements; they want to enable you to move fast and develop software as quickly as possible —not to impose any overhead.

User stories should therefore be one of the tools in your product tool box, not the only one.

PRODUCT MANAGER VS. PRODUCT OWNER

For many years, people have debated what the difference between the product manager and the product owner role is, if the roles can coexist or not, and which one should be used. This article shares my thoughts on the topic and reflects on the origin of the product owner role.

What?
As you may know, the product owner originated from Scrum, where the role is responsible for "maximising the value the product creates." [1] This sounds like a text-book product management responsibility to me. Nevertheless, the product owner is often regarded as a tactical role tasked with managing the product backlog, detailing requirements, and interacting with the development team. How come?

The confusion stems—at least partly—from the fact that Scrum is a simple framework focused on helping teams develop software. It does not cover common product management practices, such as, product strategy development, product roadmapping, and financial forecasting; and the only product management tool it offers is the product backlog.

Additionally, some approaches like SAFe employ a separate product manager and product owner role in order to facilitate scaling. Using a strategic product role and a tactical one is a common scaling technique. But calling the tactical role "product owner" is an unfortunate mistake in my mind: The SAFe product owner is not the same as the Scrum product owner! Having two different product owner roles adds to the confusion.

So What?
So why did Scrum introduce a product owner role at all? Why didn't the framework use the term product manager? When Scrum was developed in the 1990ies, product management was different from what it is today. Product managers used to do the upfront market research, product planning, and requirements definition work. They would then hand off a requirements specification to a project manager who would work with development and test to deliver the product. The product manager would return only to issue change requests or help with the product launch.

This is in stark contrast to how things are done in an agile process, where product people are required to collaborate with the development teams on an ongoing basis—without neglecting the market and the internal stakeholders.

Secondly, Scrum is applied outside the realm of product development and commercial software products. Many organisations that have adopted Scrum like banks, retailers, and media companies traditionally don't have a product management group and hence no product mangers. But they do have digital products that either help market and sell their revenue-generating offerings, such as, an online banking app, or they develop software that is used to automate business processes, increase productivity, and reduce cost.

By offering the product owner role, these organisations can start working in an agile way without the immediate need to establish a product management group and initiate an organisational change process. Instead, employees from the appropriate business units can—with some training and coaching—act as product owners . (In the long run, however, establishing a product

management function is likely to be beneficial, as I discuss in my post “Five Tips for Introducing Product Management to Your Company.”)

Now What?
So where does this leave us? My hope is that we will move past the divisive product manager-product owner debate and just talk about product people. [2]

In the short term, we should acknowledge that the product owner is a product management role. People playing the role should therefore acquire the relevant product management skills. As Marty Cagan and others, including myself, have pointed out, a two-day training course is not enough to become a competent product owner. Product management is a complex, multi-faceted discipline that takes time and effort to master.

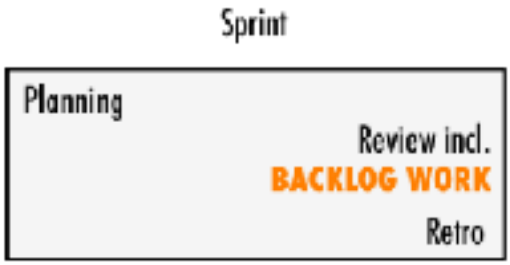
Additionally, I recommend using either the term product manager or product owner in your company and qualifying it when necessary, for instance, by employing the terms senior and junior product manager / owner and strategic and tactical product manager / owner. This reduces confusion and helps unite people.

What matters are not job roles and titles. It’s the good we do for the users and our businesses.

WHEN SHOULD PRODUCT BACKLOG GROOMING TAKE PLACE?

A well-groomed product backlog facilitates the development of a successful product: It incorporates new insights and learning, and it provides items that are ready to be implemented. But when should you work on the backlog? Before the new sprint starts or afterwards? And how can you decide which option is appropriate? In this post, I discuss four options with their benefits and drawbacks to help you make the right choice.

Option 1: In the Sprint Review Meeting
Your first option is to work on the product backlog in the sprint review meeting. Assuming that the development has developed a “done” product increment and the right people are present, you can use the attendee’s feedback to make the relevant product decisions and update the backlog, as the Scrum Guide suggests and the following picture shows.



This approach ensures that the backlog is updated before the next sprint starts. This allows you to immediately action any new insights thereby mitigating the risk of taking the product in the wrong direction. Additionally, the backlog is collaboratively worked on. This creates strong buy-in from the people participating in the sprint review meeting.

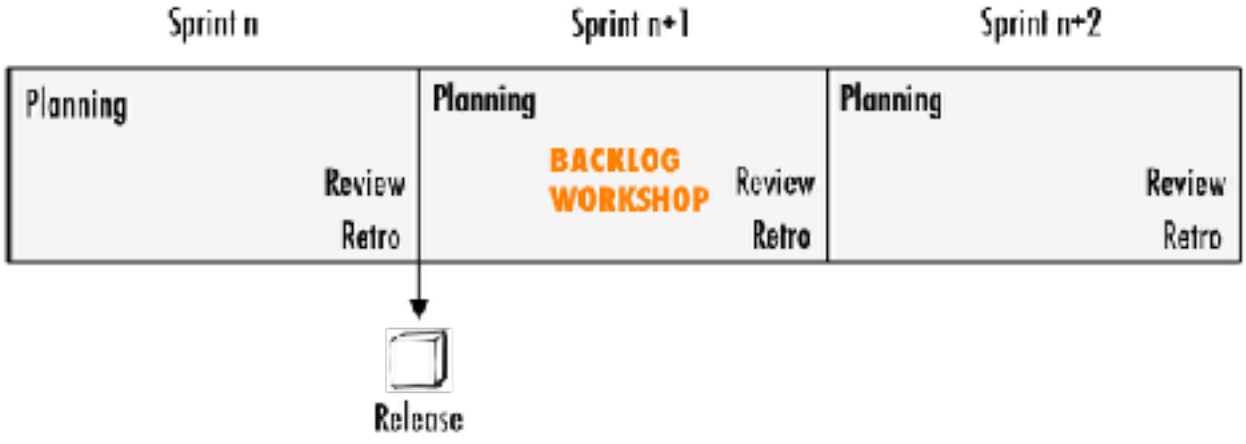
But it only works if the attendees are able to provide relevant feedback, are willing to collaborate and can quickly agree on the necessary backlog changes. What’s more, the feedback must not require further analysis or give rise to bigger backlog changes. Don’t use this option if a significant amount of uncertainty and change is present in your product backlog, for example, if you work on a new product or extend your product’s life cycle.

Option 2: In a Separate Workshop Prior to Sprint Planning
Your second option is to have a separate product backlog workshop prior to the next sprint planning meeting. The workshop should include you as the product owner, the development team, and the ScrumMaster. Collaboratively analysing the data and working on the backlog, mitigates the risk of drawing the wrong conclusions due to cognitive biases; it leverages the creativity and knowledge of the team, increases understanding and buy-in, and leads to better requirements.

As the picture above illustrates, my preference is to hold the workshop right at the beginning of the next sprint. This violates the Scrum rule that the sprint planning meeting must be the first event in every sprint. But it allows you to respond to the feedback in the subsequent sprint without rushing or dropping the sprint retrospective or working late or at the weekend. What's more, it provides the benefit of separating the data collection from the analysis: It allows you to review the feedback without the people present who provided it. It also give you more time to assess the feedback, derive the right insights, make the right product decisions, and change the product backlog accordingly. This makes it easier to deal with difficult feedback and requests, and to carry out bigger product backlog changes.

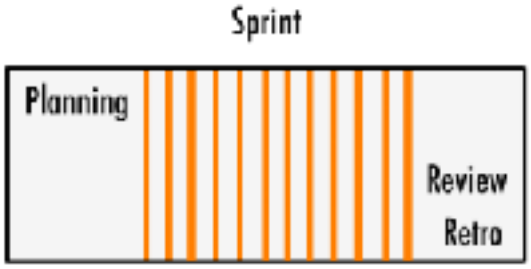
Note that this approach still assumes that you can collect the relevant feedback in the sprint review meeting, for instance, by carrying out a product demo, usability test, or solution interview. If you decide to release the product increment to (selected) users, then it is unlikely to work for you, as collecting the relevant data often takes several days in my experience. Take a look at the next option instead.

Option 3: In a Separate Workshop After Sprint Planning
If you validate your product decisions by releasing software to (selected) users, then option three may suit you. It suggests that you hold a product backlog workshop involving the development team and ScrumMaster after the next sprint planning has taken place, as the picture below shows. This approach assumes that you require several days to gather the relevant user data before you can analyse it and make the appropriate backlog changes. It also assumes that you don't have to wait for the data to arrive to decide on the next sprint goal. Instead, you can continue with the next sprint while collecting the data.



While option three allows you to validate your product quantitatively, it too has a drawback: As sprints are protected, the earliest point in time you can respond to the backlog changes is sprint+2. It is therefore advisable to start with option one and two, and use option three once the crucial risks in the backlog have been addressed. Additionally, you should focus on a different feature in sprint n+1 compared to sprint n in order to avoid the danger of building on wrong decisions.

Option 4: Continuously
If you don't need to wait until the end of the sprint before you release a new or improved piece of functionality, then option four is for you: You continuously collect data, analyse it, and change the backlog accordingly. You may want to set a side 30 minutes every morning to look at the latest data and draw the right conclusion from it, preferably with the help of the development team or some of its members.



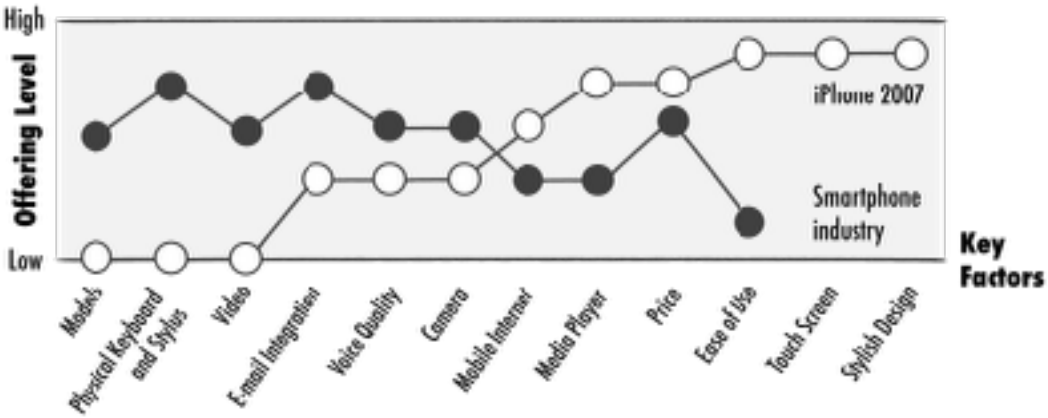
Note that option 4 assumes that you either make small incremental improvements to a product or that you run multiple overlapping experiments to test ideas for new or improved features, for example, by releasing feature fakes or MVFs.

Strictly speaking, Scrum is not well suited to support this approach. The model assumes that a cross-functional development team has to work together for several weeks to achieve a shared sprint goal and deliver a product increment. A sprint is intended to run one experiment or implement related product functionality. It's not particularly well suited to execute multiple tests and continuously release smaller enhancements and bug fixes. You might therefore consider changing from Scrum to a Kanban-based process, as I discuss in more detail in my post Is Scrum Right for Your Product?.

MAKE YOUR PRODUCT STAND OUT WITH THE STRATEGY CANVAS

Few products are ground-breaking innovations with zero competition. Chances are that alternatives for your product exist. You should therefore ensure that your product stands out from the crowd and that people have a compelling reason to choose it over competing offerings. The Strategy Canvas is a great tool to achieve this, as I explain in this post.

The Strategy Canvas
The Strategy Canvas was developed by Kim and Mauborgne, the authors of Blue Ocean Theory. It was originally intended as a business strategy tool to discover new markets. Luckily, the



canvas can also be applied to individual products, as the following example illustrates

The Strategy Canvas above ranks the first iPhone against its rivals, including the Nokia N95 and the BlackBerry Curve. The horizontal axis of the canvas captures the key factors companies compete on to provide their products. In the example above, there are twelve factors, which range from offering different models to stylish design.

The vertical axis of the Strategy Canvas describes the offering level, the degree to which the competitors offer the factors. Video, for instance, was not offered on the original iPhone but it was provided by its competitors; a camera was available on the first iPhone but it's quality was inferior compared to the competition; but its media player was better, and the iPhone offered two new factors, a touch screen and a stylish design.

By assessing the degree to which the iPhone and the competitor products fulfil the twelve factors, two lines are created. The dark one represents the value curve of the smartphone industry in 2007; the light one the first iPhone.

When comparing the two lines, we see that the they diverge. This means that the first iPhone was clearly differentiated. Apple achieved this by removing certain features, such as physical keyboard, stylus, and video; reducing some, including voice quality and e-mail integration; and improving others, for example, mobile Internet and media player. Additionally, the two new factors—the touch screen and stylish design—gave the phone a significant competitive advantage and helped Apple disrupt the mobile-phone market.

Applying the Canvas to Your Product

Before applying the Strategy Canvas, you should be clear on your product's value proposition—the main problem it solves or the primary benefit it provides—and the market segment you want to serve. I capture these pieces of information using my Product Vision Board. (The canvas therefore complements the board in my approach.) Additionally, you should know who your main competitors are.

Start creating the canvas by determining the key factors. Make sure you choose those factors that define the current standard in your market and are used to advertise and sell products, rather than the ones that favour your own product. Product reviews and test reports can help you discover the right factors, since they compare a product against the expected standard. Additionally, limit the number of factors you use to about ten. This creates focus and avoids an overly complex canvas.

With the key factors in place, rank the competing offerings and your own product taking into account the degree to which they fulfil the factor. This is not meant to be a scientific exercise but based on the information you have collected whilst identifying the key factors. Consider if a factor is fulfilled not at all, hardly, to some extent, or fully, and rank the products accordingly.

With the scores in place, represented as dots or circles, connect them to create the value curves. Think about joining up the curves of the competitors to create clarity, as I did in the Strategy Canvas above. While this carries the risk of overlooking some details, it simplifies the canvas and makes it easier to see where the main opportunities are for making your product stand out.

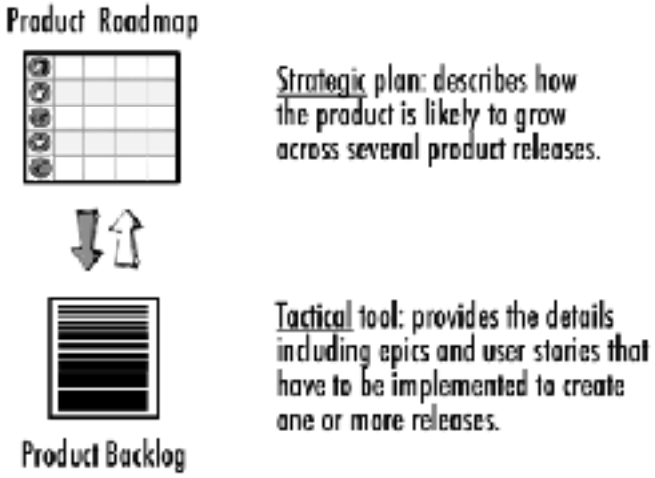
If the value curve of your product is too close to the curve of the competition, then you haven't differentiated your product sufficiently. You will subsequently find it hard to explain to your users and customers why they should choose your product. What you would like to see instead is a value curve that significantly diverges from the industry standard, like the white-dotted one in the Strategy Canvas above. This is achieved by eliminating, reducing, and raising the appropriate key factors, and by creating new ones. The Eliminate-Reduce-Raise-Create Grid can help you with this.

Use the Strategy Canvas not only for brand-new products. Regularly check if an existing product is still adequately differentiated or if the competition has caught up with it. Take the iPhone. If we consider how the smartphone market has changed since the launch of the iPhone, we see that the competitors have matched the original iPhone's features. Trying to stay ahead of the competition, Apple has introduced and raised several factors over the years, including the ability to install third-party apps, the number and size of iPhone models, the battery life, and the camera and video capabilities.

10 PRODUCT BACKLOG TIPS

Working with the product backlog can be challenging, and many product owners wrestle with overly long and detailed backlogs. This blog post provides ten practical tips that help you work with your product backlog effectively.

Tip #1: Complement your Product Backlog with a Product Roadmap
Use a roadmap to sketch the overall journey you want to take your product on. State the upcoming major releases with their goals or benefits. Then derive your product backlog from the roadmap and use the goals to discover the right backlog items. This ensures that your backlog is aligned with the product strategy, and it helps you decide which items should be added to the product backlog and which should not.



Tip #2: Focus your Backlog on the Next Major Release
Use the product backlog as a tactical tool that states the product details–including epics and user stories–that have to be implemented to deliver the next major release. This results in a concise backlog, which is comparatively easy to update and change. The longer-term growth of your product should be captured on the product roadmap.

Tip #3: Start with a Short and Sketchy Product Backlog
Particularly when you create a new product or new features and keep the lower-priority items coarse-grained. Use the user and customer feedback to decide which feature to implement, to evolve the product backlog, and to refine its items. It's OK, however, to have a longer, more detailed backlog when your product is mature and your focus is on incremental changes and bug fixes.

Tip #4. Collaborate with the Development Team

Involve the team members in the product backlog work. This allows you to benefit from their knowledge and creativity and discover technical risks and dependencies. It also increases the understanding and buy-in of the team members and results in better, clearer requirements.

Tip #5: Say No
Decline ideas and requirements that do not help you meet the release goal and move you closer to realising the product vision. This ensures that your product has a clear value-proposition and it prevents your product from getting bloated. If the idea or requirement is important but cannot be realised in the next few months, then consider adding it to the product roadmap.

Tip #6: Look beyond User Stories
While user stories and functional requirements in general are important, they are usually not enough. Also consider the user interaction, the nonfunctional qualities of your product, and the user interface and capture them in your product backlog.

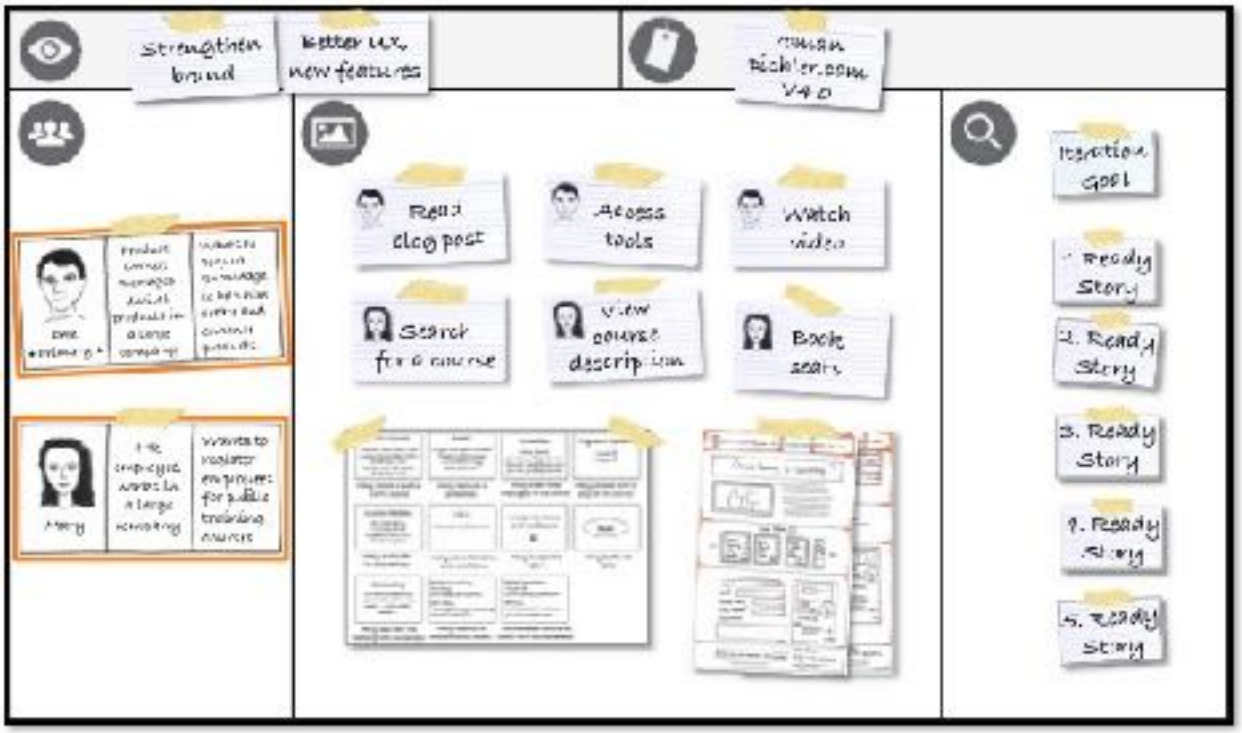
Tip #7: Prioritise your Backlog
Use uncertainty and risk to decide how soon an item should be implemented. Addressing uncertain items early on allows you to test your ideas, to fail fast, and to learn how to continue. Complement risk with cost-benefit and take into account dependencies when necessary.

Tip #8: Proactively Manage your Product Backlog
Regularly groom and refine it together with the development team. Analyse the feedback and data collected from exposing the latest product increment to the users and apply the new insights to the backlog: remove and add new items, and update existing ones. This maximises the chances of building a product that users really want and it keeps the product backlog up to date and concise.

Tip #9: Get the Backlog Ready
Break larger items into smaller ones by leveraging the insights gained from exposing product increments to the users. Ensure that the high-priority items are ready for sprint planning: the items should be clear, feasible, and testable. This facilitates a realistic commitment, and it helps the team turn the items into a product increment without having to constantly ask you during the sprint what the user story means and if there is something missing.

Tip #10: Make your Product Backlog Visible and Easily Accessible
Try a paper-based backlog and put it on the wall. Such a backlog offers several benefits:

It is clearly visible and creates transparency–assuming that it’s on the team room wall and people are colocated.
It alerts you when your backlog is becoming too big, as you will be running out of wall space.
A tool like my Product Canvas helps you structure and visualise your backlog.



If using a paper-based product backlog is not possible, employ an electronic tool that is easy to use; or consider a mixed approach with some of the items on the wall and others–like the high-priority stories–in a tool like JIRA.

http://www.romanpichler.com/blog/product-roadmap-vs-release-plan/

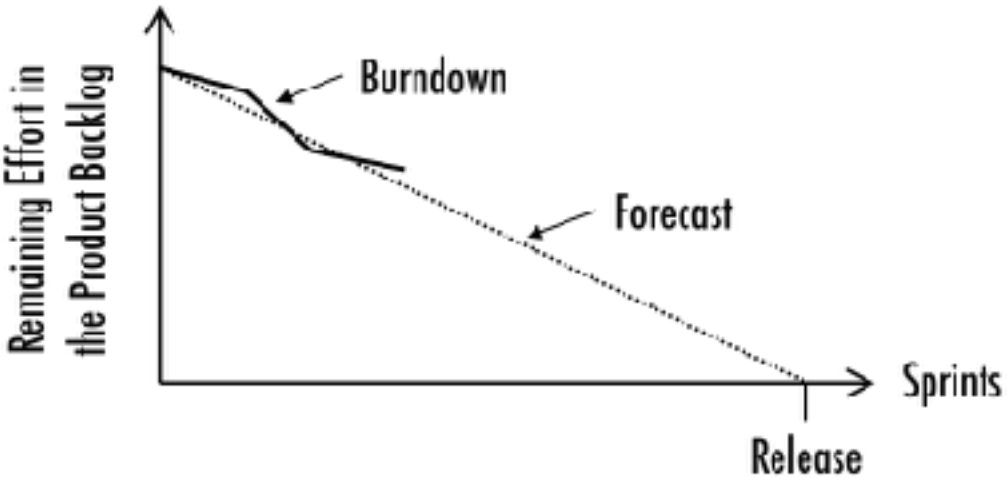
THE PRODUCT ROADMAP AND THE RELEASE PLAN

Release planning and product roadmapping are both important practices to achieve product success. But what's exactly the difference between a release plan and a product roadmap? How do the two tools fit together? This post answers these questions so you can apply the two planning artefacts effectively.

What is a Release Plan?

A release plan forecasts how a major release is developed. It's a type of project plan—albeit an agile one—and it usually covers the next three to six months. I use the term major release to refer to a version of your digital product that introduces a noticeable change, for instance, by adding or optimising functionality or enhancing the user experience, and it typically results in a new product version—think of Windows 10 or iOS 9.3, for example.

Release plans come in different shapes and sizes depending on the process used. In Scrum, the release burndown chart is the default release plan. It helps you track the progress from sprint to sprint, anticipate if the relevant product backlog items can be delivered on time and budget (or how long it will take and how much it will cost), and to make the necessary adjustments, such as, reduce or remove a feature, or add a new team member to the team. The following picture shows a sample release burndown chart.



In the chart above, the vertical axis captures the remaining effort in the product backlog required to create the next release, and the horizontal axis captures the number of sprints necessary or available to develop the release. The first data point on the chart is the estimated effort of the entire product backlog before any development has taken place. To arrive at the next data point, you determine the remaining effort in the product backlog at the end of the first sprint. Then draw a line between the two points. The burndown line shows the progress that has been made, and after a few sprints you should see a trend emerge and be able to forecast future progress. The forecast is represented by the dotted line in the chart above.

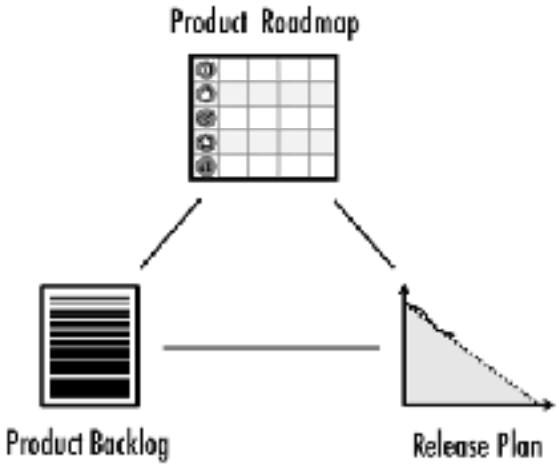
What is a Product Roadmap?
A product roadmap communicates how a product is likely to evolve across several major releases. Unlike the release plan, it is a product plan that looks beyond an individual project or release: It describes the journey you want to take your product on over the next 12 months or so —much like a roadmap helps you plan a road trip.

Product roadmaps vary in their format. I prefer to work with goal-oriented roadmaps (also called theme-based roadmaps). As their name suggests, these roadmaps focus on the goals the upcoming releases should provide. The picture below shows the GO Product Roadmap, a specific goal-oriented roadmap format that I have created. You can download the roadmap template from romanpichler.com/tools or by clicking on the following image

Let's take a quick look at the rows of the GO roadmap in the picture above from top to bottom. The first row captures the date or the time frame when a new product release should be available—for example, 1 March 2015, or first quarter 2015. As explained in my post 10 Tips for Creating an Agile Product Roadmap, I recommend using dates or timeframes on internal product roadmaps and omitting them on external ones, that is, on roadmaps that are shown to (prospective) customers. The second row states the name of the release like iOS 9.3 or Windows 10 mentioned before.

The third row is the most important part of the GO roadmap. As its name suggests, it states the goal of a release, the benefit it should provide, and the reason for creating it. Sample goals include acquiring users, improving the user experience, and removing technical debt. The fourth row lists the product's features that are necessary to meet the goal. Derive the features from the goals and ensure that they help create the desired benefits. Focus on what really counts; limit yourself to five features per release, and keep the features coarse-grained. The fifth and final row captures the metrics to determine if a release goal has been met—for example, x amount of users employ the product for at least thirty minutes per day within two weeks after the release becomes available. Stating the metrics ensures that the goals on your roadmap are specific and measurable.

How do the Release Plan and Roadmap Relate?
I find it helpful to think of the product roadmap as a high-level plan that sketches out the major stages of a road trip, including overnight stops. The release plan then states how each stage is likely to unfold. My preference is therefore to derive the release plan from the product roadmap (with the help of the product backlog) and to use the release plan to forecast how a specific release goal on the product roadmap is met. The following picture shows the relationship between the three artefacts.



Don't forget to keep the product roadmap and the release plan in synch. Bigger changes in the release plan are likely to impact the roadmap. If, for instance, the development progress is slower than anticipated, then this will not only impact the release plan but it might also affect the product roadmap.

The following table summarises the key differences between the release plan and product roadmap.

Artefact	Characteristics	Planning Horizon	Contents
Release Plan	Project plan, tactical	3 to 6 months	Product backlog items, including user stories
Product Roadmap	Product plan, strategic	12 months	Release goals, high-level features / product capabilities

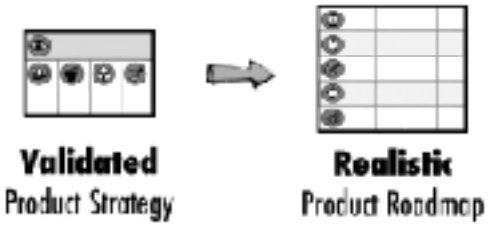
10 TIPS FOR CREATING AN AGILE PRODUCT ROADMAP
http://www.romanpichler.com/blog/10-tips-creating-agile-product-roadmap/

A product roadmap is a powerful tool to describe how a product is likely to grow, to align the stakeholders, and to acquire a budget for developing the product. But creating an effective roadmap is not easy, particularly in an agile context where changes occur frequently and unexpectedly. This post shares ten practical tips to helps you create an actionable agile product roadmap.

1 Focus on Goals and Benefits
Whenever you are faced with an agile, dynamic environment—be it that your product is young and is experiencing significant change or that the market is dynamic with new competitors or technologies introducing change, you should work with a goal-oriented product roadmap, sometimes also referred to as theme-based. Goal-oriented roadmaps focus on goals or objectives like acquiring customers, increasing engagement, and removing technical debt. Features still exist, but they are viewed as second-class citizens; they are derived from the goals and used sparingly.

To help you develop your agile product roadmap, I have created a goal-oriented roadmap template called the GO Product Roadmap. It is built on the idea that goals are more important than features, and it consists of five elements: date, name, goal, features, and metrics, as the picture below shows. You can download the template for free from romanpichler.com/tools, and you can find more information on how to use it in my post The GO Product Roadmap.

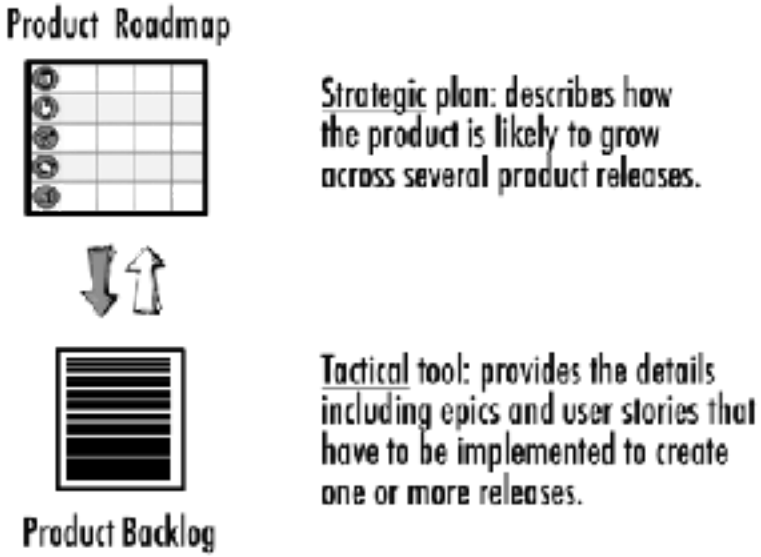
2 Do the Necessary Prep Work
Describe and validate the product strategy—the path to realise your vision—before you create your roadmap and decide how the strategy is best implemented, as the following picture illustrates.



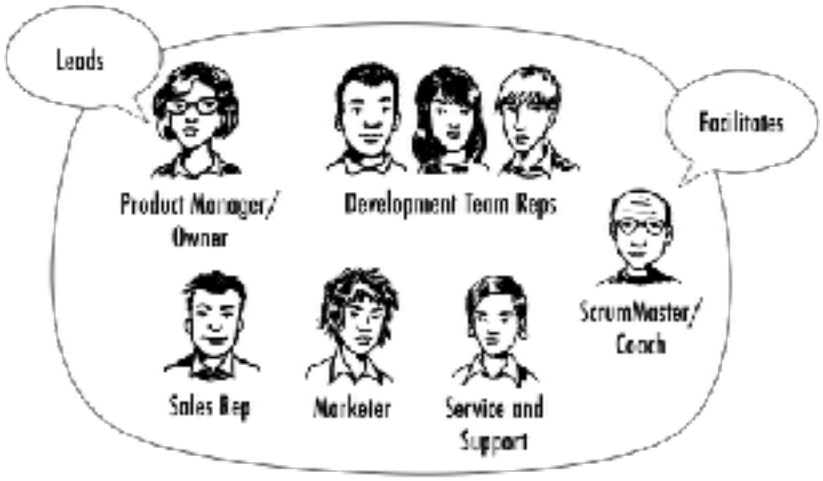
I like to use my Product Vision Board to develop a valid product strategy. The board captures the vision, the target group, the problem to be solved or the benefit to be provided, the key features of the product, and the business goals. You can download the Product Vision Board template from romanpichler.com/tools/ for free.

3 Tell a Coherent Story
Your product roadmap should tell a coherent story about the likely growth of your product. Each release should build on the previous one and move you closer towards your vision. Be clear who your audience is: An internal product roadmap talks to development, marketing, sales, service, and the other groups involved in making your product a success; and external roadmap is aimed at existing and prospective customers. Keep your roadmap realistic: Don't speculate and don't oversell your product.

4 Keep it Simple
Resist the temptation of adding too many details to your roadmap. Keep your roadmap simple and easy to understand. Capture what really matters and leave out the rest by focusing on the goals. Keep the features on your roadmap coarse-grained and derive them from the goals. The details, including the epics, user stories, scenarios and UI designs, belong in the product backlog and not on your roadmap, as the picture below shows.



5 Secure Strong Buy-in
The best roadmap is worthless if the people required to develop, market, and sell the product don't buy into it. The best way to create agreement is to collaborate with the key stakeholders to create and update the product roadmap. This allows you to leverage their ideas and knowledge and creates strong buy-in. Running a collaborative roadmapping workshop is a great way to engage everyone and create a shared product roadmap, as the following picture illustrates.



Roadmapping Workshop

6 Have the Courage to Say No
While you want to get buy-in to from the key stakeholders, you should not say yes to every idea and request. This would turn your product roadmap into a feature soup, a random collection of features. "Innovation is not about saying yes to everything. It's about saying no to all but the most crucial features," said Steve Jobs. Use your vision and product strategy to make the right decisions. Have the courage to say "no". Remember: Collaboration requires leadership.

7 Know When to Show Dates
Some people recommend to never show dates on a product roadmap, others always include them. I recommend to use dates or timeframe on an internal roadmap that coordinates the work carried out by the internal stakeholders, such as, the development team, marketing, sales, and support. This is particularly important for date-driven products like smartphones that must be ready for Christmas sales or a travel app that has to be updated before the summer holidays start. But when you use an external roadmap that is shown to customers and users and often used as a sales tool, then I recommend not showing any dates or timeframes but sequencing your releases and possible employing a now-next-later grid to order them.

8 Make your Roadmap Measurable
When using a goal-oriented roadmap, ensure that every goal is measurable. This allows you to tell if you have met the goal or not. If your goal is to acquire customers, for example, then ask

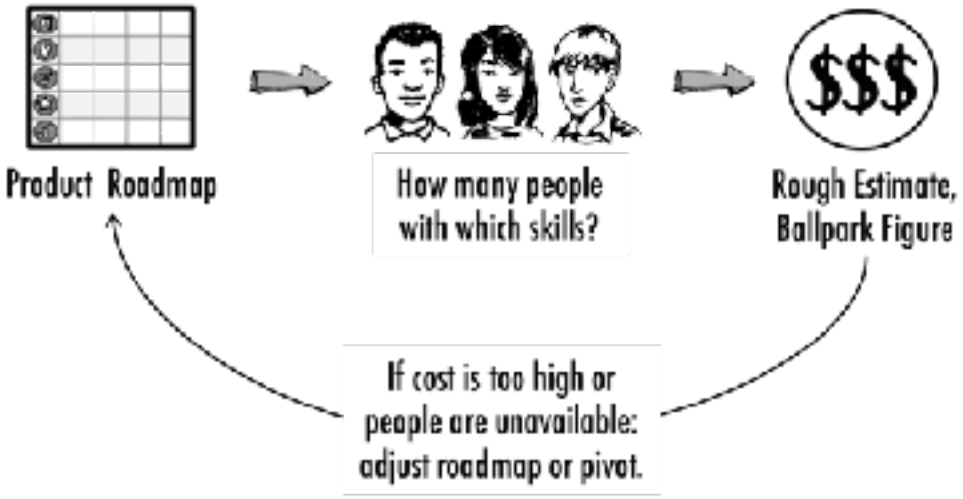
yourself how many new customers should be acquired; or if your goal is to reduce technical debt, determine how much of the bad code should be removed or rewritten.

If you don't state a target, it will be hard to tell if you have met the goal or not. Make sure, though, that you state a realistic target, and that the goals on your roadmap are realistic. Then select the metrics that will help you determine if a goal has been met and if a release has delivered the desired benefit.

9 Determine Cost Top-Down

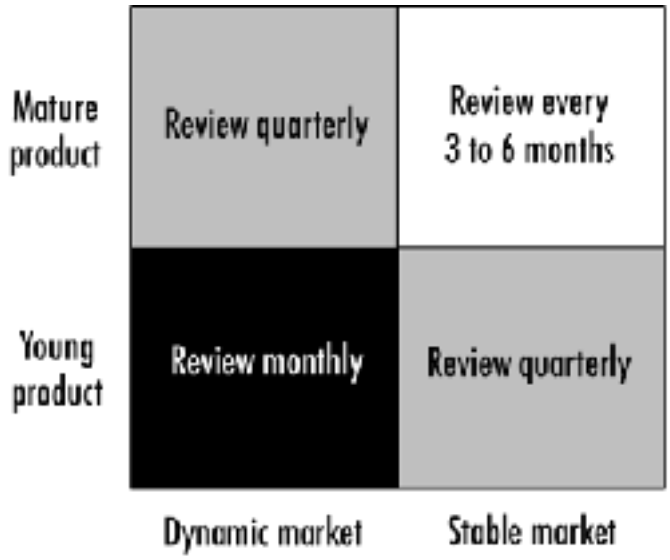
Whenever your product is new, young, or changing, I recommend that you do not attempt to determine the development cost bottom-up but rather top-down. It's virtually impossible to derive the right epics and user stories from the roadmap features, get correct estimates from your team, and accurately anticipate the velocity and the rate of change in the product backlog. Even if you manage to make it work, you will end up with an overly long and complex product backlog that is difficult to adjust and maintain. What's more, it can take days—and in some cases weeks—to turn the features into well-defined requirements and to come up with detailed estimates.

Instead, determine how many people with which skills are likely to be required to create the desired releases on the roadmap. Draw on your experience of developing similar products or previous versions of the same product; consider whether enough people with the right expertise are available in your company, or if you will have to hire or contract people. This should give you an indication of the likely labor cost required. Then add the cost for facilities, infrastructure, materials, licenses, and other relevant items. Carry out this exercise together with the development team.



10 Regularly Review and Adjust the Roadmap

Last but not least: If the environment you're in is agile, then change is likely to occur. You should therefore regularly review and update your product roadmap—between every four weeks to every three months depending on how young your product and how dynamic the market is.



http://www.romanpichler.com/blog/goal-oriented-agile-product-roadmap/

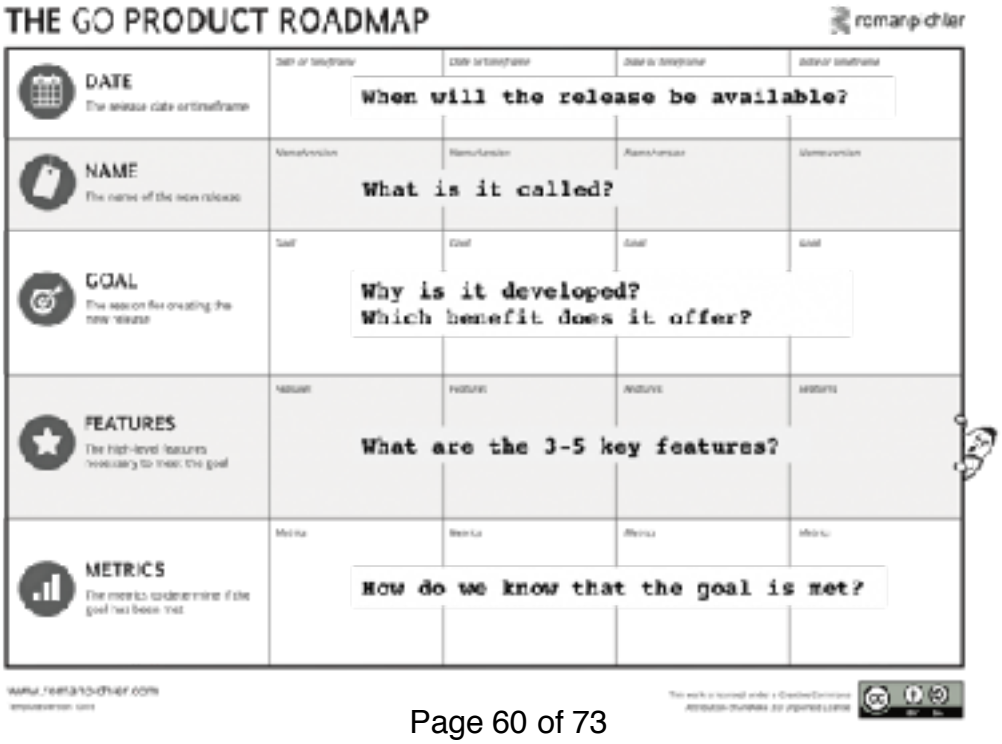
THE GO PRODUCT ROADMAP – AN AGILE PRODUCT MANAGEMENT TOOL

Product roadmaps are an important product management tool. But applying them effectively can be challenging. Roadmaps are too often focused on features, and product managers spend too much time getting the stakeholders to agree on when which features will be developed. This post introduces a new product roadmap template: the GO product roadmap, a goal-oriented roadmap that combines goals and features in a novel way.
A product roadmap is a high-level, strategic plan, which describes how the product is likely to develop and grow over the next months. This creates a continuity of purpose, and it helps product managers and product owners acquire funding for their product; it aligns stakeholders and facilitates prioritisation; it makes it easier to coordinate the development and launch of different product; and it provides reassurance to the customers (if the product roadmap is made public).

Unfortunately, I find that many product managers and product owners struggle with their roadmaps, as they are dominated by features: There are too many features, and the features are often too detailed. This turns a roadmap into a tactical planning tool that competes with the product backlog. What's more, the features are sometimes regarded as a commitment by senior management rather than part of a high-level plan that is likely to change.

Faced with this situation, I have developed a new goal-oriented agile roadmap — the GO product roadmap, or "GO" for short. GO is based on my experience of teaching and coaching product managers and product owners, as well as using product roadmaps in my own business.

The GO Product Roadmap Explained
The following pictures shows what the GO product roadmap looks like. You can download a PDF template by simply clicking on the picture.



The first row of the GO roadmap depicted above contains the date or timeframe for the upcoming releases. You can work with a specific date such as 1st of March, or a period such as the first or second quarter. I find dates particularly valuable on internal roadmaps. If you use an external one, you may want to remove the row or use loose timeframes, such as, in the first six months of 2014.

The second row states the name or version of the releases, for instance, iOS 7 or Windows 8.1.

The third row provides the goal of each release, the benefit the new release should provide. Sample goals are to acquire or to activate users, to retain users by enhancing the user experience, or to accelerate development by removing technical debt. Working with goals shifts the conversation from debating individual features to agreeing on desired benefits making strategic product decisions. The development team, the stakeholders, and the management sponsor should all buy into the goals.

The fourth row provides the features necessary to reach the goal. The features are means to an end, but not an end in themselves: They serve to create value and to reach the goal. Try to limit the number of features for each release to three, but do not state more than five. Refrain from detailing the features, and focus on the product capabilities that are necessary to meet the goal. Your product roadmap should be a high-level plan. The details should be covered in the product backlog, and commitments should be limited to individual sprints.

The last row states the metrics, the measurements that help determine if the goal has been met, and if the release was successful. Make sure that the metrics allow you to measure if and to which extent you have met the goal. Consider how long it takes after the release has become available to determine to determine goal attainment. For instance, if your goal is to acquire 1000 new users, then you may have to wait a few days to understand if your goal has been met.

A Sample GO Product Roadmap
To illustrate how the GO template can be applied, imagine we are about to develop a new dance game for girls aged eight to 12 years. The app should be fun and educational allowing the players to modify the characters, change the music, dance with remote players, and choreograph new dances. Here is what the corresponding GO roadmap could look like:

	1st quarter	2nd quarter	3rd quarter	4th quarter
	Version 1	Version 2	Version 3	Version 4
	Acquisition: Free app, limited in-app purchases	Activation: Focus on in-app purchases	Retention	Acquisition: New segment
	• Basic game functionality • Multiplayer • FB integration	• Purchase dance moves • Create new dances	• New characters and floors • Enhanced visual design	• Street dance elements • Dance competition
	Downloads: top 10 dance app	Activations, downloads	Daily active players, session length	Downloads

While the roadmap above will have to be updated and refined at a later stage (particularly the metrics), I find it good enough to show how the product may evolve and make an investment decision.

When creating your GO roadmap make sure you determine the goal of each release before you identify the features. This ensures that the features do serve the goal. Filling in the roadmap template from top to bottom and from left to right works well for me.

8 TIPS FOR COLLABORATING WITH DEVELOPMENT TEAMS

The development team is a key partner for every product manager and product owner: the team designs and builds the actual product. But it's not always easy to effectively guide and work with the team. This post shares eight tips to make your collaboration with the development team even more effective, thereby increasing the chances of creating a successful product.

Manage the Product, not the Team

Focus on your job as the product manager or product owner, and manage the product, not the team. Provide guidance on the product, including its market, value proposition, business goals, and key features. Care about the team members. But let the ScrumMaster or coach tackle people, process, and organisational issues; and let the development team figure out what needs to be done to implement the user stories and other product backlog items.

It's a common mistake to step in and take on the ScrumMaster role if there is no ScrumMaster or if the individual is struggling to do a good job. While this may hep you in the short-term, it's going to hurt you in the long run. Taking on too many responsibilities means spreading yourself too thinly: something will have to go. You will either neglect some of your product responsibilities or put your health at risk. Neither is desirable. (For more guidance on the relationship between the product owner and ScrumMaster see my post Every Great Product Owner Needs a Great ScrumMaster.)

Treat the Team as an Equal Partner

Remember the Golden Rule? Treat others how you want to be treated. The team members are not your resources but the people who create your product. If your relationship with the team is poor, then your product is likely to suffer. Assume that the team members want to do their best. Respect their UX/UI and technology decisions and their right to determine how much work can be done. Be honest and open. Provide constructive feedback and share your concerns. But don't tell people how to do their job and refrain from assigning tasks. Development teams should manage their own work (using a sprint backlog or Kanban board). If the team struggles, then it's the ScrumMaster's job to help the team—not yours (as discussed above).

Help the Team See the Bigger Picture

Developing a successful digital product requires more than technical knowledge. It's virtually impossible to develop the right solution without understanding the product context, including who the customers and users are, what value the product creates for them, what makes the product stand out, and how it benefits the business. You should therefore help the team acquire the relevant market and domain knowledge—for instance, by involving the team in research and validation work, inviting them to join you when you visit customers— and ensure that they are aware of the product strategy and product roadmap as well as the business goals and KPIs. This will not only lead to better technical decisions and a better product. It also eases your workload: understanding the bigger picture allows the team to help create user stories and to support you in managing the product backlog.

Involve the Team in Product Decisions

While you own and manage the product, the development team should understand and support important product decisions. The best way to achieve strong buy-in is involving the team members in the decision-making process. This also leverages their creativity and knowledge and is likely to lead to better decisions. There are a number of techniques that help you achieve strong buy-in including the following ones:

Creating shared goals: a vision, release goals, and sprint goals everyone agrees on;
Involving the team members in research and validation activities, for example, observing users or building MVPs and jointly analysing the resulting data;
Engaging the team in developing and updating the product roadmap;
Collaborating on the product backlog: prioritising items and creating user stories.
Be aware that collaboration requires leadership. As the person in charge of the product, you should be open and collaborative but decisive at the same time. Aim to build consensus with the development team, but don't shy away from difficult conversations. Don't settle for the smallest common denominator and have the courage to make a decision if no agreement can be achieved: Great products are not based on weak compromises.

Spend Enough Time with the Team but Don't Neglect your Other Duties
Make time to collaborate with the team in order to work on user stories, answer questions, and participate in meetings. If you are not available or difficult to reach, you are not guiding the team. In the worst case, people get fed up with trying to get hold of you or waiting for an answer and stop consulting you. Consequently, you may end up with a product that requires extra rework or has features that cannot be released.

If, however, you feel overwhelmed by the team's questions, then coach the team to help people see the bigger picture and involve the team in product backlog management and user story creation. This will allow them to carry out their work autonomously and it reduces the amount of questions you have to answer during the sprint while the team is implementing the stories.

As important as it is to make enough time for the team, don't neglect your other product management duties, such as, engaging with users, working on the product strategy and roadmap, and managing the stakeholders. If you become too team-focused, your product is likely to suffer.

Expect High Standards but Don't Push People
Hold the team accountable and expect that people do a good job–that commitments are kept and agreements respected, that sprint goals are delivered, that the team adheres to the definition of done and creates software that works, is documented, and tested. But recognise that software development can be challenging and that human beings make mistakes. Don't be mad with the team if the sprint goal is missed once. But don't accept it if the team repeatedly doesn't deliver what was promised. Use the sprint retrospective to investigate the causes and explore how you can help, for instance, by creating smaller user stories or better acceptance criteria.

Don't force work on the development team and don't ask people to carry out more tasks than they can realistically cope with. Otherwise the team may become demotivated and start to take shortcuts like compromising quality and neglecting documentation. In the worst case, people will fall ill or leave. Instead, involve the team in setting a meaningful sprint goal that provides motivation and guidance for the team and respect the team's right to decide how much work can be done. This creates a sustainable pace and keeps your team motivated.

Give the Team Room to Experiment and Learn
In order to generate value, a product has to offer something new; it has to innovate to a greater or lesser extent. To help your product create value in the future, the team requires time to learn

their skills and to investigate new technologies and tools. But this is unlikely to happen if you expect that people work new features all the time. You should therefore give the team enough room to experiment with new ideas and acquire new knowledge. Some teams use gold cards to allocate time in a sprint for experimentation and learning; others use hack days. Whatever works for your team, help people prepare for the future. This will benefit your product and the team morale.

Fully Participate in the Meetings (or Don't Show up)
This might seem like a trivial piece of advice, but I've seen my fair share of product people who half-heartedly participated in meetings with the development team. Therefore, come prepared to a meeting and fully participate—silence your phone, put away your laptop and tablet—or don't attend.

In a Scrum context, the two most important meetings for product managers and product owners are usually sprint planning and sprint review. You should always aim to be present at these meetings and do the necessary prep work, such as, prioritising the product backlog and refining user stories for sprint planning or inviting the right people and selecting the right product validation technique for the review meeting. But don't facilitate the sessions. Let the ScrumMaster take on this role.

HOULD PRODUCT OWNERS BE SERVANT-LEADERS?

Being an effective product owner or product manager requires leadership: the development team and stakeholders need guidance and direction to collaborate and achieve product success. But as product people, we usually don't have any authority over the individuals; we can't tell people what to do. Can servant-leadership—leading others by serving them—help product owners and product managers guide and direct people?

What Is Servant-Leadership?

Servant-leadership means that "one wants to serve, to serve first. Then conscious choice brings one to aspire to lead," writes Robert Greenleaf, the creator of the servant-leadership model. Servant-leaders want to "make sure that other people's highest priority needs are being served" so that they "become healthier, wiser, freer, more autonomous."

Sounds crazy? To me, servant-leadership means leading from the heart: recognising that the people we work with are first and foremost human beings. Caring for them and building strong relationships with them are prerequisites for achieving great things together. If we don't care for the people we want to lead, they are unlikely to trust and follow us.

Caring for the followers, the people we work with, is present in other leadership models, too. Transformational leadership urges leaders to show genuine concern for the needs and feelings of their followers, for example, and Goleman's leadership styles recognise affiliative leadership as an approach that puts people first. In Scrum, servant-leadership is regarded as the default leadership approach embodied by the ScrumMaster.

How Can Servant-Leadership Benefit Product Owners?

Servant-leadership encourages us to reflect on our motivation to be product leaders. Why do we want to lead others? Is it to gain power, status, success, or money? Is it to succeed together and help the people we lead grow and develop?

There is nothing wrong with gaining respect as well as a bonus or pay rise. But if these motivators dominate our thinking, we are in danger of no longer caring for the people we lead but seeing them as tools to advance our personal ambitions. This negatively affects our relationship with them and reduces their willingness to trust and follow us.

Servant-leadership also helps us question our approach to achieving success. Is the end—a successful product—more important than the means—how we got there? Do we expect that people just do their job, function, perform, and deliver? Are working extra hours and weekends normal to get a product release out? Or do we take an interest in the people we work with, show kindness to them, and encourage the developers to go home when they look tired and worn out?

Don't get me wrong—I am no utopian. I know that great products are the result of hard work. But if we want sustained success and a work environment that is healthy and conducive to creative work, then we must care for the people we work with and lead—the development team and the stakeholders. This starts with taking a real interest in others, making an effort to be present and engage properly.

Next time you are tempted to tell the dev team that you expect them to work extra hours, stop and reflect. Check if there isn't room for thinning out a feature or hiring an expert that would accelerate the progress. And when this annoying programmer complains about the user stories again or the pushy sales guy reiterates that the product strategy is wrong, take a deep breath.

Recall that we all have healthy and unhealthy traits and remind yourself of the individual's positive qualities. Then kindly but clearly share your thoughts.

THE PRODUCT CANVAS

This post introduces my Product Canvas, a simple but powerful tool that helps you create a product with a great user experience and the right features. It combines Agile and UX by complementing user stories with personas, storyboards, scenarios, design sketches and other UX artefacts. It's designed to work with Scrum, Lean Startup, and Business Model Generation. The canvas supports Lean UX by combining user centred-design and agile techniques

A Sample Canvas

The best way to understand the Product Canvas is to look at an example. Image that we want to develop a game that helps children enjoy music and dancing. A canvas for such a game could look like the one below.

contains the product name, the product (or release) goal and the metrics to measure if the goal has been met. The first bigger section states two personas characterising the target users and customers with their needs. The next section sketches important aspects of the product using epics to describe the product's functionality, a mock-up to capture the user interface design, a storyboard to illustrate the user interaction, and a constraint card to express the platform for which the game is developed. The section on the right provides a goal for the next sprint and the details necessary to reach the goal.

The Sections Explained

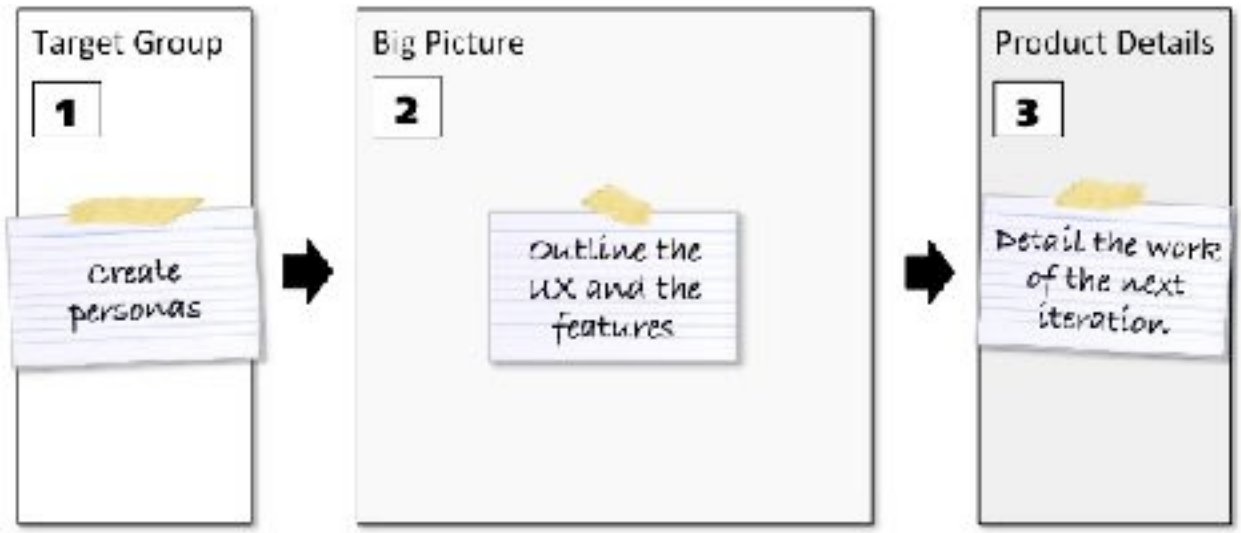
As you have probably noticed, the Product Canvas combines form and function, a structure together with suggested techniques. The following diagram and the text below the sections of the canvas.
the product. The Goal is the product or release goal, the objective that should be met, for instance, to acquire, activate or retain users. If you

use the GO product roadmap than you can simply copy the relevant goal stated on the roadmap. The Metrics provides the measure to determine if the goal has been met, for instance, number of downloads or daily active sessions. If you use the GO product roadmap than just copy the relevant roadmap metrics. The Target Group describes the target customers and users as personas. The section explains who we believe is likely to use buy and use the product and why. I discuss personas in more detail in my post A Template for Writing Great Personas. Choose one primary persona – the persona you mainly create the product for. Employing a primary persona helps you make the right prioritisation decisions and create a product with a great user experience. Your primary persona should be at the top of the building block to signal its importance. The Big Picture describes what is takes to meet the persona goals. It captures the user journeys, and the visual design required to create the desired user experience. As its name suggests, it wants to describe your product holistically at a high-level. The section is similar to the outline of a book: It captures the contents without discussing the details. Scenarios, storyboards, workflow diagrams, and story maps are great techniques to describe the user journeys on the Big Picture. Each journey shows how a persona interacts with the product and the steps the individual has to take to meet a goal. The product functionality on the Big Picture is best captured as epics, which are big and coarse-grained user stories. Epics allow you to describe your ideas without having to commit to the details. This saves time, and it makes it easier to update the canvas with new insights. Constraint stories help you capture the nonfunctional requirements that impact the user experience and the software architecture. You can capture your visual design ideas on the Product Canvas as design sketches, mock-ups, screen-shots, and photos. The Big Picture design artefacts should focus on the critical design aspects of your product—for instance, the design of selected screens or pages. None of these techniques are mandatory, of course. They rather provide you with a starting point. Choose those techniques that are appropriate for your product. Use additional ones as it suites your needs. The Product Details provide a goal for the next iteration and just enough

implementable items to reach the goal, for instance, to address a risk and to acquire relevant knowledge, or to complete a feature. Depending on the goal, I use different techniques to capture the implementable items. For goals that require coding, ready stories are very helpful. These are small, detailed stories that feed the next cycle and that help create a product increment or minimal viable product (MVP). They are derived from the epics, and are necessary to reach the sprint goal. Make sure you write acceptance criteria for your ready stories. Order the implementable items from one to n, for instance, first, second, third, and so on, to maximise the chances that you reach your goal.

Putting the Users First

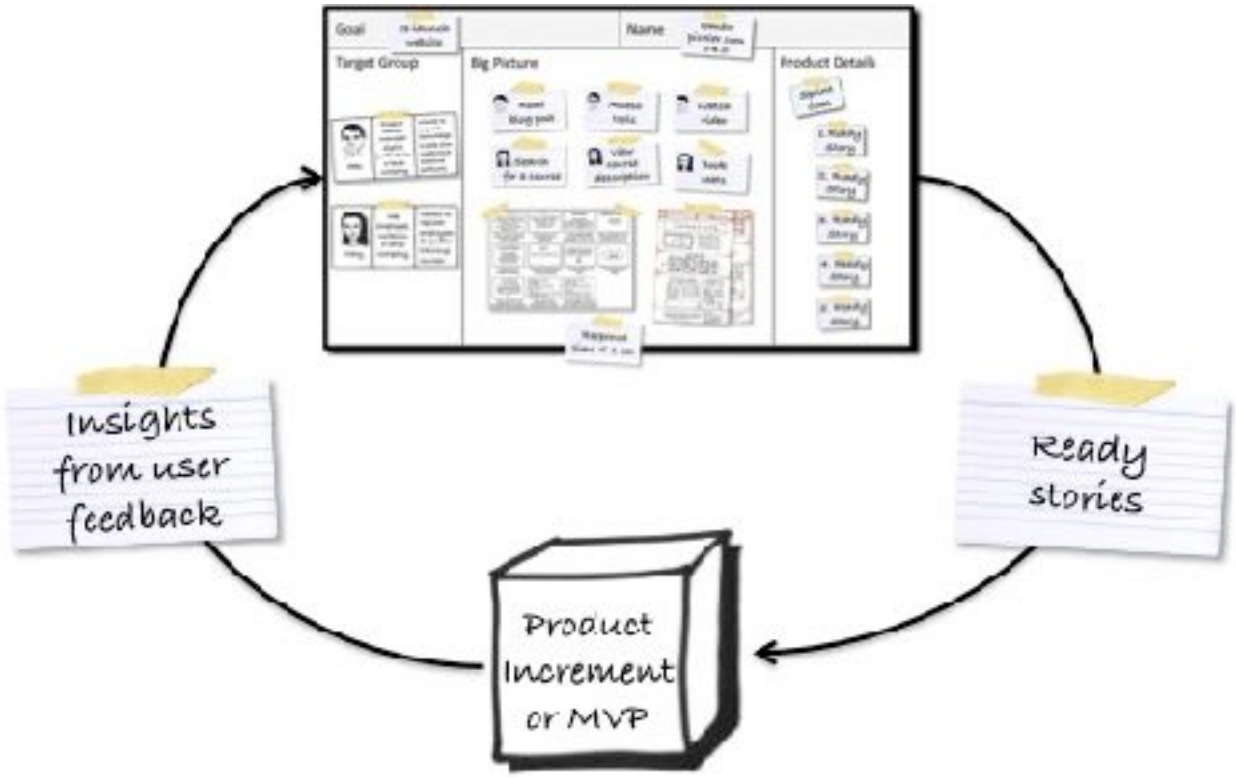
The canvas is designed so that the information flows from left to right starting with the personas. This puts the user at the center of the development effort, and it ensures that you develop a product that is beneficial and desirable.



The scenarios, storyboards, epics, design sketches, and constraints describe the future product, and the ready stories ensure that there are implementable items. I explain in more detail how you can create you canvas in my post "The Product Canvas Creation Workshop".

Learning and Emergence

The biggest challenge when developing a new product is to deal with uncertainty and lack of knowledge. We may not know, for instance, if there is enough demand for the product, or how users will interact with the product. The Product Canvas is designed as a learning tool: to sketch our initial ideas, to get enough stories ready for implementation, and to adapt and refine the content based on the insights gained. The following picture illustrates this cycle.

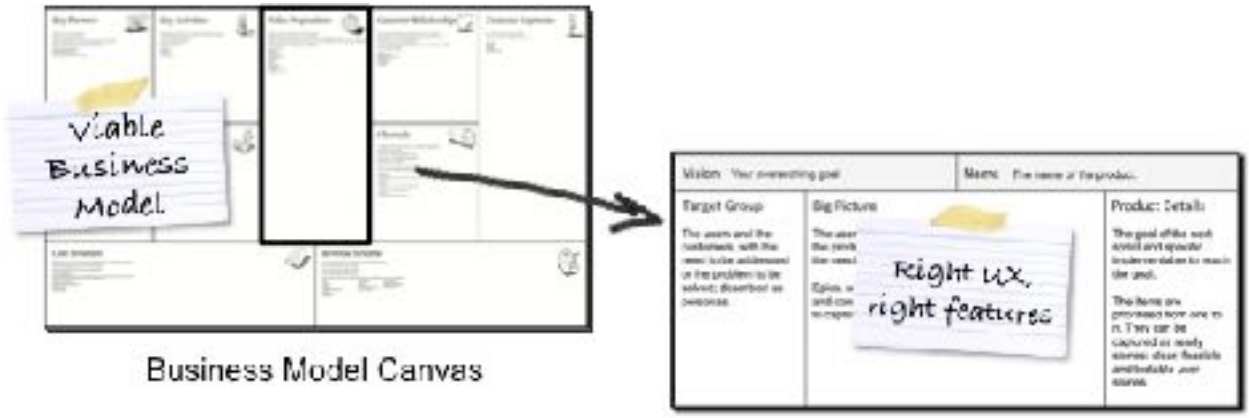


Consequently, you should expect that your canvas changes as you learn more about the users and customers, and how to best address their needs. It's common to deal with bigger changes involving clearing out and refilling one or more canvas sections including the section on personas.

The Business Model

The Product Canvas describes the target group and the product features, but not the business model including the revenue sources and the cost structure. While I have intentionally kept the canvas focussed, I have

designed it to be compatible with the Alexander Osterwalder's and Yves Pigneur's Business Model Canvas. You can use the two canvases together, as the following picture illustrates.



Business Model Canvas

Product Canvas

As the picture above suggests, I use the Business Model Canvas to capture the market and value proposition at a high-level, and I state the details for a specific product on the Product Canvas.

Tools

I like to work with a physical Product Canvas placed on the office wall, as this has three main benefits: First, it ensures that the relevant information is visible to the product owner and the team. Second, working with simple, yet effective tools such as paper cards and paper sheets facilitates effective collaboration. Third, having to work with limited wall space creates focus and prevents capturing everything that might be relevant. To create your own paper-based canvas, download and print out my free Product Canvas template. If you require an electronic canvas, then consider using a wiki, or try Jolien Coenraets' Google Drive template. There are also several software tools, which allow you to create a Product Canvas including the BMFiddle Product Canvas, the ProdPad, and the Canvas Model Design iPad app. If you work with JIRA, I suggest you keep the personas and the big picture in your wiki or project management tool, and manage the product details in your JIRA board.

Wrap-up

The Product Canvas brings together the key pieces of information necessary to create a new product: the users and customers with their needs, the product's functionality and design, and the user interaction. It's intended to be a collaborative tool that helps you state your ideas and assumptions, test them, and integrate the insights you gain. Try it out, and let me know how the canvas works for you! I'd love to learn how using the canvas has worked for you.