SHIP IT

HOW AGILE PRODUCT MANAGERS CAN BUILD BETTER PRODUCTS

ProductPlan

SHIP IT

HOW AGILE PRODUCT MANAGERS CAN BUILD BETTER PRODUCTS

TABLE OF CONTENTS:

WHY WE WROTE THIS BOOK

When the team at ProductPlan originally set out to write a book about agile product management, we realized we were in a unique position. Not only have we had the fortune of engaging with thousands of product managers over the past few years, but we also have several people on our team with long-time experience in agile software development.

More than 15 years ago, I wrote my first requirements document for one of the first SaaS products. At the time, the requirements were detailed in a thick monolithic Word document that covered exactly what the new solution needed to be, along with every one of the "must have" product features.

Only then, after I thought I had documented everything, did software development begin.

What we eventually developed over the subsequent months was a close approximation of what I had written. But during development, a lot changed. Decisions were made, features pared back. The vision of the product I had written didn't exactly match the reality and difficulty of developing software. Fortunately, the product sold to millions of business customers and remains a commercial success today.

Since then I've helped launch several products using agile software development practices. The process is a stark contrast to the typical "waterfall" development that is still in use by many companies today.

At ProductPlan, we believe that agile development practices—combined with what we call "agile product management"—can bring products to market faster and more successfully.

With this book we hope to give you some practical advice for developing and managing amazing products.

We hope you enjoy it.

Jim Semick
Co-Founder, ProductPlan
www.productplan.com

1

UNDERSTANDING AGILE THROUGH THE LENS OF A PRODUCT MANAGER

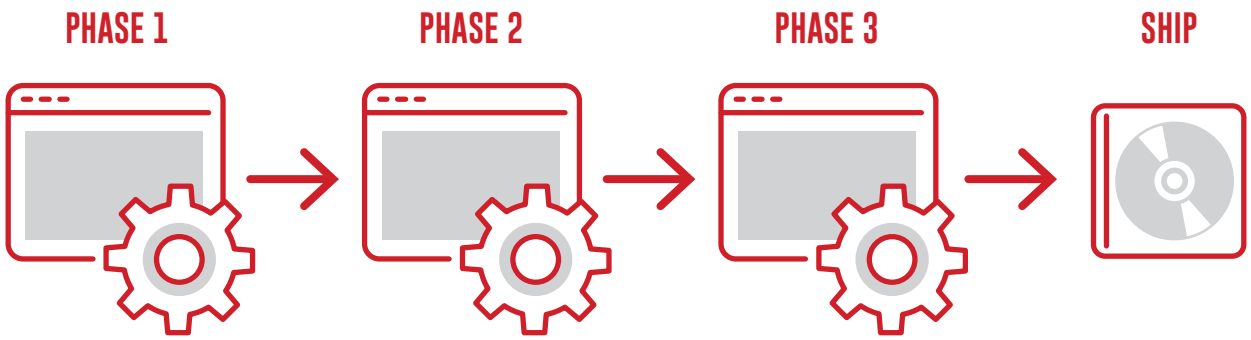
1 UNDERSTANDING AGILE THROUGH THE LENS OF A PRODUCT MANAGER

THE STORY OF AGILE

Over the past few decades, software development practitioners around the world have helped drive a movement we now refer to as "agile." But it wasn't long ago that other traditional approaches to software development were the status quo.

Predecessors to agile typically involved long planning and development cycles that followed a rigid sequential order. This kind of linear approach is often referred to as "waterfall" development. Waterfall dates back to the 1950s—more or less the beginning of software engineering.

With the methodologies of the pre-agile era, every single detail had to be planned ahead. Several cycles were spent gathering detailed requirements and producing extensive documentation before a single line of code was written.



Typically waterfall organizations only update their roadmap once a year or so. And in previous decades, it wasn't uncommon for software development teams to plan release cycles that spanned multiple years!

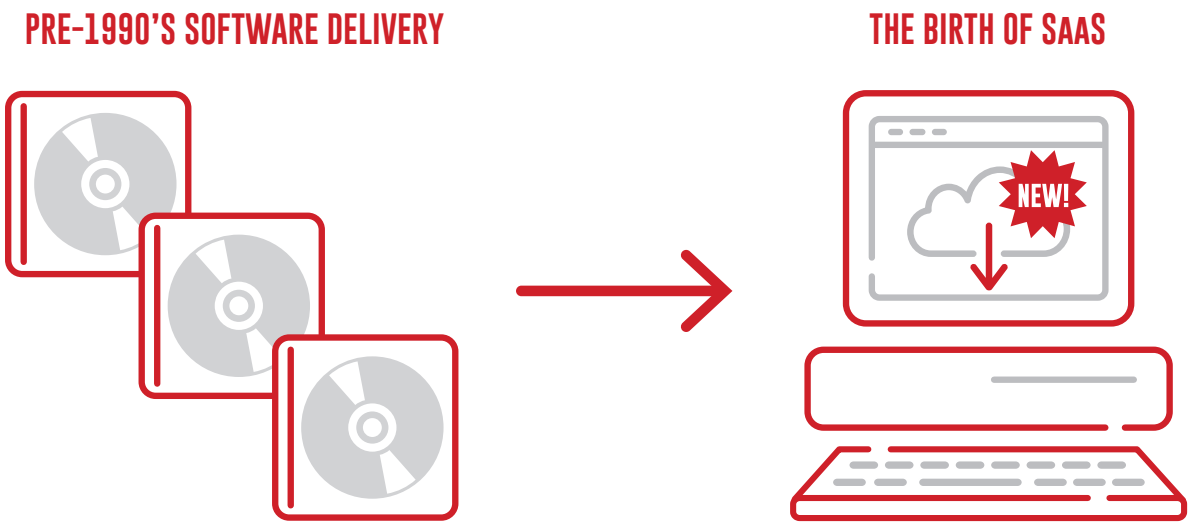
Without effective ways to quickly validate ideas within the market, many new products were developed heavily based on assumptions. (And we all know what they say about assumptions.)

In the absence of early prototypes, MVPs, and early-stage customer feedback to shed light on potential product-market-fit, traditional software development processes carried a lot of risk.

We owe a lot to waterfall, which can now be considered the grandfather of software development methodologies. It served product managers well for decades. But as with any field, innovation was inevitable. As the landscape of software development shifted dramatically towards the end of the 20th century, so too did approaches to managing it.

1990'S: NEW SOFTWARE DELIVERY OPTIONS

During the 1990's, web-based software and SaaS business models began to make their debuts. This movement away from physical software sales meant the long development and release cycles of the past were soon to be on their way out. The ability to deliver software updates to users immediately enabled development teams to start deploying principles like iterative testing thanks to immediate access to actionable feedback from real customers.



2001: THE AGILE MANIFESTO

In February 2001, 17 software development practitioners gathered at a ski resort in Utah. They were there to ski. They were there to relax. They were there to eat and drink. But most importantly, they were there to lament, pontificate, and solve problems.

Despite having widely varying opinions on the right way to approach software development, the crew agreed on at least one thing: the status quo was not working. There was increasing need for an alternative to documentation-driven and heavyweight software development processes.

Out of their gathering in Utah that winter came The Agile Manifesto, a brief document built on 4 values and 12 principles for agile software development.

It's important to note that agile in itself wasn't born then. Its creators, and many other software development practitioners, had long been applying various agile values and principles piecemeal. But The Agile Manifesto made concrete the ideas that had been permeating the software development world for the last decade or so.

APPLYING THE AGILE MANIFESTO TO PRODUCT MANAGEMENT

THE 4 AGILE VALUES

The agile mentality is guided by 4 overarching values which differentiate it from traditional software development processes.

4 AGILE VALUES

"Through this work we have come to value:
Individuals and interactions over processes and tools
Working software over comprehensive documentation
Customer collaboration over contract negotiation
Responding to change over following a plan"

INDIVIDUALS AND INTERACTIONS OVER PROCESSES AND TOOLS

No matter how well-researched your process and high-tech your tools are, it's the team you work with and the way you work together that determines success. Your team and their ability to communicate effectively and efficiently is more valuable than the processes they follow or the tools you use.

This isn't to say that agile philosophies discourage formalized processes or tools. Both can be helpful in providing structure for your team and facilitating interactions. But at the end of the day, they come second.

After all, processes and tools are worthless if your team can't communicate. But put a smart, motivated team up to a task without any processes or tools to manage the project and chances are they'll find some way to get it done.

AGILE VALUES PROMOTE GIVING INDIVIDUALS THE TRUST AND AUTONOMY THEY NEED TO WORK EFFECTIVELY.

WORKING SOFTWARE OVER COMPREHENSIVE DOCUMENTATION

Traditional product development processes often required extensive documentation before a single line of code was written. Under the agile philosophy, getting software in the hands of customers is the highest priority. After all, how are you going to improve your product if you don't get it out in the wild and collect feedback from real users?

While this value highlights the importance of shipping software over letting documentation be a bottleneck, it's important to note that documentation in itself is not a bad thing...as long as you don't overdo it.

CUSTOMER COLLABORATION OVER CONTRACT NEGOTIATION

The agile philosophy highlights the importance of customer-centric product development practices over product-centric approaches. While contracts will always have their place in business, a list of the things you're offering your customer is no replacement for actually communicating with them about what their needs are and where their challenges are.

Traditional product-centric processes allowed contracts to dictate what was delivered in the end, which left a lot of room for mismatched expectations. The agile philosophy (and many of the formalized processes that have come out of it) encourage building a continuous customer feedback loop into development cycles.

Under the agile philosophy, customer collaboration begins early in the development process and happens frequently throughout. This culture of close collaboration with real customers helps product people ensure they're delivering effective, useful

solutions to customers. When you talk to customers often and build feedback into your development process, you reduce risk and eliminate guesswork.

RESPONDING TO CHANGE OVER FOLLOWING A PLAN

An important benefit of the agile methodology is that it encourages frequent reviewing and retooling of current plans based on new information that the team is continually gathering and analyzing. The product roadmap, then, is no longer a static document, but a dynamic strategy. Product managers in agile environments will need to learn to present their dynamic roadmaps to stakeholders in a transparent manner that reflects the likelihood of change based on new learnings.

In other words, the agile methodology lets a product team adjust its priorities and plans whenever doing so makes strategic sense. These teams do not get stuck in an outdated plan simply because they have committed to seeing it through.

THE 12 PRINCIPLES OF AGILE SOFTWARE DEVELOPMENT

AGILE PRINCIPLE 1

"OUR HIGHEST PRIORITY IS TO SATISFY THE CUSTOMER THROUGH EARLY AND CONTINUOUS DELIVERY OF VALUABLE SOFTWARE."

The best ways to ensure you make customers happy while continuously delivering valuable software are to ship early, iterate frequently, and listen to your market continually.

Unlike traditional approaches to product development, which have notoriously long development cycles, agile principles encourage minimizing the time between ideation and launch. The idea is to get a working product in the hands of customers as soon as possible. Doing this successfully means product managers are able to quickly get a minimum viable product (MVP) out and into the world and use it to get feedback from real customers. This feedback is then fed back into the product development process and used to inform future releases.

HOW IT LOOKS IN PRACTICE:

- Product teams use minimum viable products and rapid experimentation to test hypotheses and validate ideas.
- Frequent releases help fuel a continuous feedback cycle between customer and product.
- Shipped and done are not the same thing. Instead of releasing a "finished" product, iterations continue to make incremental improvements to it based on customer and market feedback.

AGILE PRINCIPLE 2

"WELCOME CHANGING REQUIREMENTS, EVEN LATE IN DEVELOPMENT. AGILE PROCESSES HARNESS CHANGE FOR THE CUSTOMER'S COMPETITIVE ADVANTAGE."

In the world around us, change is the only constant. Agile principles and values support responding to these changes rather than moving forward in spite of them. Previous approaches to product development were often change averse; detailed, well-documented plans were made before development began and were set in stone regardless of new findings. Agile principles support observing changing markets, customer needs, and competitive threats and changing course when necessary.

HOW IT LOOKS IN PRACTICE:

- Product teams are guided by high-level strategic goals and perhaps even themes below those goals. The product department's success is measured against progress toward those strategic goals rather than by delivery of a predefined featureset.
- Product constantly has its ear to the ground monitoring the market, customer feedback, and other factors which could influence product direction. When actionable insight is uncovered, plans are adjusted to better serve customer and business needs.
- Product strategy and tactical plans are reviewed, adjusted, and shared on a regular cadence to reflect changes and new findings. As such, product needs to manage the expectations of executive stakeholders appropriately and ensure they understand the "why" behind changes.

AGILE PRINCIPLE 3

"DELIVER WORKING SOFTWARE FREQUENTLY, FROM A COUPLE OF WEEKS TO A COUPLE OF MONTHS, WITH A PREFERENCE TO THE SHORTER TIMESCALE."

Agile philosophy favors breaking a product's development into smaller components and "shipping" those components frequently. Using an agile approach, therefore—and building in more frequent mini-releases of your product—can speed the product's overall development.

This agile approach, with short-term development cycles of smaller portions of the product, results in less time spent drafting and poring over the large amounts of documentation that characterizes waterfall product development. More importantly, this frequent-release approach creates more opportunities for you and your teams to validate your product ideas and strategies from the qualified constituencies who see each new release.

HOW IT LOOKS IN PRACTICE:

- Agile development cycles, often called "sprints" or "iterations" break down product initiatives into smaller chunks that can be completed in a set timeframe. Often this timeframe is between 2 and 4 weeks which truly is a sprint if you consider the marathon-like development cycles waterfall teams often follow.

- Another popular alternative to agile sprints is continuous deployment. This method of shipping software frequently works less in terms of predetermined time boxes and more in terms of simply deciding what to do and doing it.

AGILE PRINCIPLE 4

"BUSINESS PEOPLE AND DEVELOPERS MUST WORK TOGETHER DAILY THROUGHOUT THE PROJECT."

Communication is a critical component of any project or team's success, and agile principles essentially mandate that it's a daily event. It takes a village to raise a child they say, and that applies to product as well.

A successful product requires insight from the business and technical sides of an organization which can only happen if these two teams work together consistently. Regular communication between business people and developers helps improve alignment across the organization by building trust and transparency.

HOW IT LOOKS IN PRACTICE:

- Cross-functional agile product development teams include product people. This means that product is represented on the development team and bridges the gap between technical and business aspects of the product.
- Daily update meetings, or standups, are one technique many agile shops use to put this principle in practice and keep everyone connected.

AGILE PRINCIPLE 5

"BUILD PROJECTS AROUND MOTIVATED INDIVIDUALS. GIVE THEM THE ENVIRONMENT AND SUPPORT THEY NEED, AND TRUST THEM TO GET THE JOB DONE."

A key part of the agile philosophy is empowering individuals and teams through trust and autonomy. The agile team needs to be carefully built to include the right people and skill sets to get the job done, and responsibilities need to be clearly defined before the beginning of a project. Once the work has begun, however, there's no place in agile for micromanagement or hand holding.

HOW IT LOOKS IN PRACTICE:

- Product must clearly ensure engineering understands strategy and requirements before development starts. This means not only sharing user stories with the cross-functional team but also the bigger picture outlined in the product roadmap.
- Product is not responsible for explaining how something should be built. They need to share what and why, but it's the delivery team's job to determine the how. Furthermore, during sprints, product does not micromanage outcome, instead they make themselves available to answer questions and provide support as needed.

AGILE PRINCIPLE 6

"THE MOST EFFICIENT AND EFFECTIVE METHOD OF CONVEYING INFORMATION TO AND WITHIN A DEVELOPMENT TEAM IS FACE-TO-FACE CONVERSATION."

With so many distributed or remote development teams these days, this principle gets a bit of critique. But at the root of it, effective communication with developers means getting these conversations out of Slack and email and favoring more human interaction (even if done by video conference calls). The overall objective behind this principle is to encourage product people and developers to truly communicate in real time about the product, requirements, and the high-level strategy driving those things.

HOW IT LOOKS IN PRACTICE:

- Daily standup meetings
- Collaborative backlog grooming sessions
- Sprint planning meetings
- Frequent demos
- Pair programming

AGILE PRINCIPLE 7

"WORKING SOFTWARE IS THE PRIMARY MEASURE OF PROGRESS."

Proponents of the agile philosophy are quick to remind us that we're in the business of building software, and that's where our time should be spent. Perfect, detailed documentation is secondary to working software. This mentality pushes to get products to market quickly rather than let documentation or an "it's not done until it's perfect" mentality become a bottleneck. The ultimate measure for success is a working product that customers love.

HOW IT LOOKS IN PRACTICE:

- Designing and releasing minimum viable features rather than fully-developed feature sets means thinking first and foremost about the smallest things we can ship to start getting customer feedback and validate as we continue to build software.
- A fail fast mentality means moving forward even in times of uncertainty and testing ideas rapidly.
- Ship software often: a useful product now is better than a perfect one later.

AGILE PRINCIPLE 8

"AGILE PROCESSES PROMOTE SUSTAINABLE DEVELOPMENT. THE SPONSORS, DEVELOPERS, AND USERS SHOULD BE ABLE TO MAINTAIN A CONSTANT PACE INDEFINITELY."

Keeping up with a demanding, rapid release schedule can be taxing on a team, especially if expectations are set too high. Agile principles encourage us to be mindful of this, and to set realistic, clear expectations. The idea is to keep morale high and improve work-life balance to prevent burnout and turnover among members of cross-functional teams.

HOW IT LOOKS IN PRACTICE:

- Before every sprint, careful consideration is given to the amount of work that can be committed to. Development teams don't overpromise on what they can deliver. Effort estimations are a common practice in setting output expectations for development teams.
- Everyone agrees on what will get done during a sprint. Once a sprint has begun, no additional tasks are to be added except in rare cases.
- Product managers should act as gatekeepers to reduce the noise from other stakeholders and to avoid squeezing in additional unplanned work during an ongoing sprint.
- Product people should do their part in promoting a sense of psychological safety across the cross-functional team that encourages open communication and freely flowing feedback.

AGILE PRINCIPLE 9

"CONTINUOUS ATTENTION TO TECHNICAL EXCELLENCE AND GOOD DESIGN ENHANCES AGILITY."

While the agile philosophy encourages shorter cycles and frequent releases, it also puts emphasis on the importance of keeping things neat and tidy so they don't cause problems in the future. Product managers often forget about this aspect of development because they mostly don't spend their days wading through their products' codebases, but it is still of the utmost importance to them.

HOW IT LOOKS IN PRACTICE:

- The team needs to be cognizant of technical debt and the technical debt implications of any new features or initiatives added to the backlog. Developers and product need to work together to understand if and when technical debt is acceptable.
- On a regular basis, product will need to allocate development resources to refactoring efforts. Refactoring cannot be an afterthought. It needs to be an ongoing consideration.

AGILE PRINCIPLE 10

"SIMPLICITY—THE ART OF MAXIMIZING THE AMOUNT OF WORK NOT DONE—IS ESSENTIAL."

You've probably heard of the 80/20 rule—the concept that you can usually get 80% of your intended results with just 20% of the work. Agile principles encourage thinking this way; doing the things that can have the most impact. In a product management context this means having a laser sharp focus on organizational objectives and making some cutthroat prioritization decisions. Agile principles discourage building merely for the sake of building by emphasizing the importance of being strategic and building with purpose.

HOW IT LOOKS IN PRACTICE:

- Product managers need to make very focused product decisions and closely align product strategy with organizational goals while being extremely picky about which user stories and features actually make the cut. Using prioritization techniques to prioritize initiatives by effort and predicted impact is one way product teams can apply this agile principle to product development.
- The short sprints that agile is characterized by present many opportunities for rapid testing and experimentation, which can help reduce uncertainty around whether initiatives will truly have the predicted impact. Using experiments to validate ideas before building them up to spec is a great way to weed out bad ideas and identify good ones.

AGILE PRINCIPLE 11

"THE BEST ARCHITECTURES, REQUIREMENTS, AND DESIGNS EMERGE FROM SELF-ORGANIZING TEAMS."

In traditional software development methodologies, you'll often see pyramid shaped teams where management makes key decisions for contributors. Agile principles suggest the use of self-organizing teams which work with a more "flat" management style where decisions are made as a group rather than by a singular manager or management team. The concept ties into agile's value of teams and interactions over processes and tools, and the intent behind the concept is to empower teams to work together as they need to.

HOW IT LOOKS IN PRACTICE:

- Self-organizing teams are autonomous groups within the organization who take control and responsibility over their respective projects and have ownership of those areas. Different organizations practice this principle differently. Spotify, for example uses "product squads" to practice this.

AGILE PRINCIPLE 12

"AT REGULAR INTERVALS, THE TEAM REFLECTS ON HOW TO BECOME MORE EFFECTIVE, THEN TUNES AND ADJUSTS ITS BEHAVIOR ACCORDINGLY."

If you're truly living by agile principles, there is no place for "we can't change because we've always done it this way." Just like we're always learning new things about our customers and markets, we're also learning from the processes we're using to learn those things. Agile is not about following a strictly-defined process for every sprint and release, it's about continuous improvement. And that continuous improvement must also extend to processes and teams.

HOW IT LOOKS IN PRACTICE:

- Experimentation and testing is not limited to the product only. Agile teams are encouraged to experiment with their processes. You may think you're already doing something well only to experiment with a revised version of the process and discover an even more effective method. Experimenting with your process and team is just as important as experimenting with the software you're building.
- Regular retrospectives provide opportunities for the team to discuss what went well, what didn't go so well, and where the process can be tweaked to improve things in the future. They're an excellent place for product managers and product owners to learn if they are communicating effectively with developers and giving them the support they need before, during, and after sprints.
- Another consideration to make regarding this principle is that, in order to practice it effectively, you need to create a culture of trust and transparency that encourages openness and frequent sharing of feedback.

AGILE IS A MENTALITY

While we're all for solving problems and giving structured, actionable advice, we'd like to point out one very important thing about agile before we delve into specifics: Agile is not prescriptive.

Notice how The Agile Manifesto doesn't outline any specific processes, procedures, best practices, or must-have roles. That's intentional. A rigid framework for following so-called "agile best practices" would be a direct contradiction of agile values and principles.

AGILE VALUE 1

Individuals and interactions over processes and tools.

The creators of The Agile Manifesto didn't set out to build a rigid framework or a methodology, they set out to build a philosophical mindset for software development. Over the years, countless adaptations of that mindset have appeared. Several formally documented and widely publicized frameworks for agile processes are widely used today.

POPULAR AGILE FRAMEWORKS

- eXtreme Programming (XP)
- Scrum
- Dynamic Systems Development Method (DDSM)
- Feature Driven Development (FDD)
- Adaptive Software Development (ASD)
- Crystal
- Lean Software Development (LSD)
- Disciplined Agile (DA)
- Scaled Agile Framework (SAFe)

Despite the broad selection of formalized agile approaches out there, you should remember one thing: There is no one-size-fits-all approach to agile. It's worth noting that out of those teams who do choose to use one of the many formalized agile frameworks, the majority of teams end up adapting it to their own needs.

Just like the first version of your software rarely functions perfectly, don't expect any formalized agile framework to work seamlessly right out of the box. You and your team should be learning and adapting every day.

Finally, it's important for you, your team, and your organization as a whole to understand that adopting a framework for agile processes, such as Scrum or eXtreme programming (XP) is only part of your journey to agile product development.

To truly be agile, you and your team must learn to think agile. Use the values and principles to guide your thinking. Fail fast. Embrace change. Never stop learning. And remember, you are never "done."

2

THE AGILE PRODUCT DEVELOPMENT TEAM

2 THE AGILE PRODUCT DEVELOPMENT TEAM

WHO BELONGS ON A CROSS-FUNCTIONAL AGILE TEAM?

In agile development, your team is everything. For best results, clearly define teams, roles, and responsibilities before work begins. But, be careful. Ensure you have a balanced team. If you include too many team members, or include people with the wrong skills, you risk slowing development or spinning off track. And if you exclude necessary team members, you'll face different but equally serious problems.

YOUR CROSS-FUNCTIONAL PRODUCT TEAM SHOULD INCLUDE ONLY THOSE ABSOLUTELY ESSENTIAL TO EACH SPRINT.

The ideal cross-functional team includes everyone deemed necessary to shepherd the product through development, but no one else. This is worth underscoring. You may find that people from different parts of your company want to be part of the agile product development team. And they often have the best of intentions. But always remember: Intentions don't matter. Results matter.

The VP of sales might believe they should share guidance based on what sales is learning from prospects. A marketing executive may also think they could contribute valuable insight during development. You might even agree with these executives. But they don't belong on the agile product development team.

Incorporating outside knowledge and perspective into your roadmap is always advised, especially under the agile philosophy. But, the last thing you want to create is a "too many cooks in the kitchen" situation on your agile team. Instead, part of your role as a product person is to be a liason for stakeholders not included on the agile team.

AGILE PRINCIPLE 5

Build projects around motivated individuals.
Give them the environment and support they need, and trust them to get the job done.

As you know, there are several agile frameworks, and many of them call for different team structures. Ultimately, you need to structure your agile team based on your needs. Over time as you learn new things, your team structure can (and should) evolve.

To give you a starting point for what a cross-functional agile team may look like, here are a few key players we see in most agile product development teams:

PRODUCT OWNER

The product owner bridges the gap between product strategy and development. They are usually responsible for the product backlog, organizing sprints, and are expected to be available to answer questions and guidance to developers as needed. We will further address the role and responsibilities of a product owner or other "dedicated product people" in the context of an agile team in the next section.

- PRODUCT OWNER
- SCRUM MASTER, TEAM LEADER OR PROJECT MANAGER
- DELIVERY TEAM
- QA RESOURCES

SCRUM MASTER, TEAM LEADER, OR PROJECT MANAGER

This point person is responsible for understanding the big development picture of each sprint. They are responsible for delegating tasks appropriately to ensure the right developers are working on the right tasks and that the team is always fully deployed and not idle. The Scrum Master role often involves working with the product owner to translate epics, stories, and other items on the sprint list into actionable tasks for developers.

DELIVERY TEAM

The delivery team consists of the folks who are in charge of executing the plan. Typically your delivery team will include the engineers and designers needed to build out the items involved in each sprint.

QA RESOURCES

It's easy to forget to include QA resources on your agile team, but they play a critical role in the process. These folks are responsible for testing the items in development before they're presented to the product owner in the sprint demo. QA resources ensure new code is continuously tested as the product is developed.

SHOULD QA AND DEVELOPMENT BE SEPARATE ROLES?

While some teams have separate development and QA staff, separating these roles is not necessary.

> ***"QA is a critical function for every team, but it does not need to be staffed by someone with QA in their title. The idea is that the team, rather than the individual, owns quality."***
>
> — **Nick Kim**
> *Software Engineer at ProductPlan and Certified Scrum Master*

STAKEHOLDERS: HONORARY MEMBERS OF THE AGILE TEAM

As we briefly discussed earlier, it's wise to keep cross-functional teams as lean as you can. This means executives and other stakeholders within your organization rarely belong on the team itself. Product serves as a liaison for various stakeholders including:

- Customers
- Prospects
- Board members
- Executives
- Other internal stakeholders

THE ROLE OF PRODUCT IN CROSS-FUNCTIONAL AGILE TEAMS

What is the role of product in an agile team anyway? Before we delve into the role as it directly relates to cross-functional agile teams, let's look at the big picture.

The role of product within an organization consists of both high-level strategic and detailed tactical components.

TACTICAL	STRATEGIC
■ Backlog grooming and fleshing out user stories ■ Making yourself available to your team and answering their questions ■ Attending sprint meetings and daily standups ■ Reviewing software as it is built and providing just-in-time feedback to engineers	■ Understanding market changes and competitive landscape ■ Working with stakeholders to understand high-level objectives for product and getting alignment from them on broader product strategy ■ Researching new markets ■ Looking at long-term strategic plans ■ Competitive analysis ■ Owning product vision and strategy/roadmap

In order to be successful, you need to have someone covering all of the bases. You cannot afford to only focus on one half of this equation. In the following sections of this chapter we'll look into best practices for the side of the product role that entails working with the cross-functional team. We'd like to point out that despite this role being largely tactical, embracing agile is not a reason to skip out on the strategic side of things. We'll address the role of strategy in the next chapter.

Every agile development team should include a product person who acts as a liaison between product strategy and the tactical side of product development. In some cases, this will be a product manager, in others, it will be a separate dedicated person. Some agile methodologies require that these roles be served by two separate people, while others don't. Some agile methodologies even have different names for this role such as Business Analyst (DDSM), Customer (XP), or even simply "product person."

Most commonly, we refer to this role as the product owner.

WHAT IS A PRODUCT OWNER?

The product owner: It's one of the most hotly debated roles in all of product management. There's even a debate as to whether the role belongs in product management at all.

Since its inception as part of the Scrum framework for software development, the role of a product owner has taken on many different and conflicting definitions. If you peruse the Internet for an explanation of the product owner role, you'll find descriptions such as the following:

DESCRIPTION 1:

A product owner is a tactical member of the product development team who attends the daily Scrum meetings and prioritizes the backlog, to ensure the developers are working as efficiently as possible and on the right items.

DESCRIPTION 2:

A product owner is a strategic role responsible for representing the interests of the customer for the development team, to ensure that there is always a user advocate involved in development meetings.

DESCRIPTION 3:

A product owner is in reality a product manager assigned to oversee sprints and to be accessible to the development team if they need help or have questions.

So, which definition is correct? The perhaps unsatisfying answer is that there is no universally accurate job description for a product owner. Some companies treat the role as tactical and task-focused—essentially a development ringleader moving the team through its to-do list.

In other companies, the product owner is viewed more strategically. In this environment, the product manager—who is perhaps too busy with all of the other responsibilities of being a product manager—communicates their strategic vision to the product owner, who then works closely with development to ensure they carry out that vision correctly.

While there are structured approaches to deciding who plays this role and what their specific responsibilities are, we'd like to remind you of an important part of the agile philosophy.

AGILE VALUE 1

Individuals and interactions over processes and tools.

At the end of the day, it doesn't matter how you break up the roles and the responsibilities of product within your organization (or what you decide to call these roles) as long as expectations are set and needs are met.

WHAT DO AGILE DEVELOPMENT TEAMS EXPECT FROM PRODUCT?

Rather than debating what constitutes the best relationship between the roles of product manager and product owner, let's look instead at requirements. What support does a cross-functional agile team need from product and how can product people ensure they're meeting the needs and expectations of their agile team? How can a product person help ensure their cross-functional team is successful?

AGILE PRINCIPLE 5

Build projects around motivated individuals.
Give them the environment and support they need,
and trust them to get the job done.

Regardless of what you call them, agile teams need a dedicated product person. This person needs to educate developers and confirm they understand the strategy before they begin building. They should make themselves readily available to answer their questions, especially during sprints. This is where the necessity for a so-called "product owner" comes from.

Problem is, many product managers, even great ones, often misunderstand or neglect this part of the role when placed on an agile team. They might perform phenomenally in carrying out their other responsibilities—communicating strategy to stakeholders, overseeing budgets and resources, synthesizing user feedback and competitive intelligence to create product roadmaps, etc.

But when it comes to the day-to-day responsibilities of real product ownership, many product managers simply fall short. It can be risky to let this role fall through the cracks in any company, but it is particularly dangerous in an agile development environment.

That's because the less you behave as a member of your development team, and the longer you go without communicating with them, the less input you'll have over what they're building, and the more likely they are to drift away from your strategic vision for the product.

To be truly effective in the role, a product owner must make a commitment to the development team that they will be available to them at all times during their sprint (or whatever format their development process takes). That means attending all

of their meetings, even the daily standups. It means being available to review and discuss all of the user stories on the team's to-do list. And it means always being available at any point during development to answer questions.

It's helpful to always think of the dedicated product owner or the product manager who is serving that role as a member of the agile development team—not an outsider who occasionally communicates with that team.

TRUE PRODUCT OWNERS WORK CLOSELY WITH DEVELOPERS TO CONFIRM THEY UNDERSTAND THE STRATEGY BEFORE THEY BEGIN BUILDING.

Here's a quick way to determine how well you're meeting your obligations as a true product owner. Ask yourself if the following statements describe you:

- I think of myself as part of the product development team. I'm not simply a product owner who writes specs, delivers them to our developers, and then waits for them to build something. I am an active member of that team, from the initial product kickoff meeting, through launch and beyond.
- I attend and actively participate in all sprint planning sessions.
- I attend and actively participate in all sprint demos. (In fact, our development team won't hold these demos without me.)
- I am always available to my team for questions or discussions about the details of the product they're coding.
- I review every story (which usually must include a mock-up) before it enters a sprint.

One way to know that you're succeeding in your role as a product owner, then, is that it will feel like you might be over-communicating with your development teams. The good news is, this behavior is encouraged within the agile philosophy.

AGILE PRINCIPLE 4

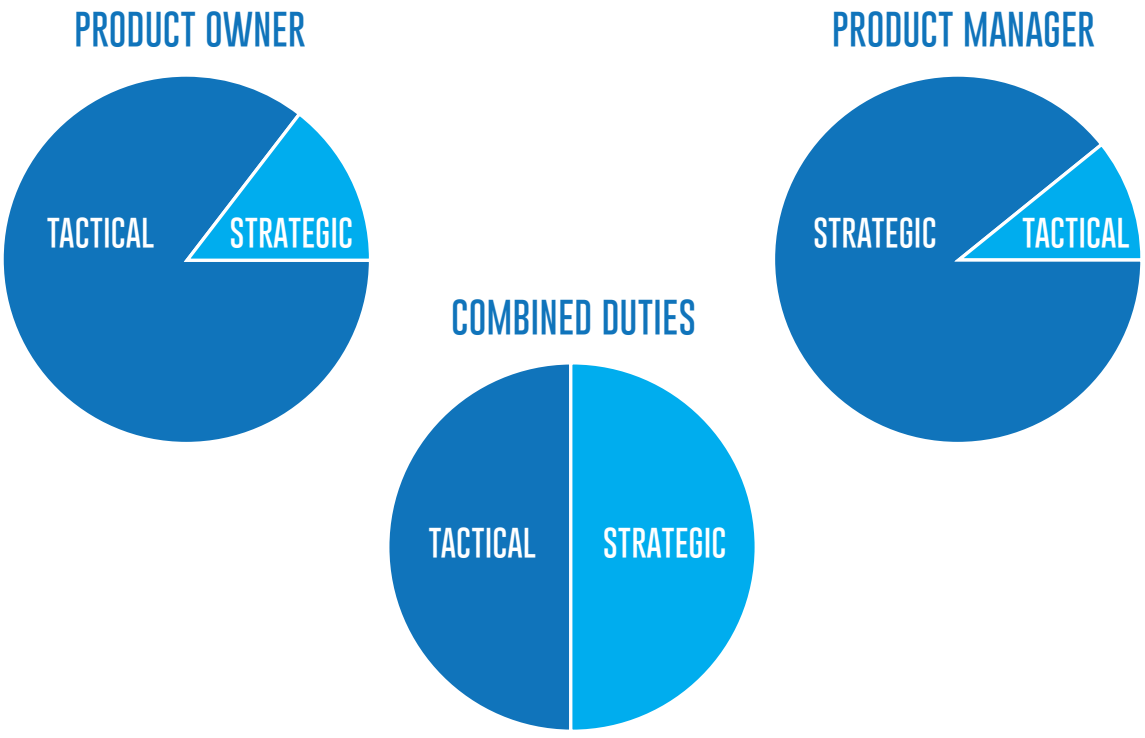
Business people and developers **must work together daily** throughout the project.

WHEN TO SEPARATE PRODUCT MANAGER AND PRODUCT OWNER ROLES

Taking on the product owner role is time consuming. Really time-consuming. If you add all of the standard product owner responsibilities to your already full plate of product management tasks, you'll find it's a lot to handle.

If you are a product manager with a company that's relatively small, or you manage only a single product (or a small family of similar products), you might be able to perform double-duty as your development team's product owner.

But because product management encompasses a far more sweeping set of responsibilities, you simply might not have the time to be the product owner as well—or at least not do so as effectively as your development team will need you to.



If you're managing a handful of large, complex products, each with its own dedicated team of developers, you probably won't be able to make yourself as accessible to all of those dev teams as a true product owner should. There's just too much else to do in your role as a product manager—communicating with your executive stakeholders; conducting market research; meeting with your marketing, sales, and customer success teams; etc.

How do you know when it might be time to hire a dedicated person for the product owner role? Oftentimes it becomes an issue of bandwidth; if you're unable to fulfill your all-important role as the product owner in addition to your other responsibilities, it may be time to consider bringing in reinforcement.

AGILE PRINCIPLE 12

At regular intervals, the team reflects on **how to become more effective**, then tunes and adjusts its behavior accordingly.

There are a few clues that might suggest you're not quite filling the product owner shoes and may need to either refocus your efforts or bring in a dedicated product owner as support. Ask yourself if these statements sound familiar:

- When I review what my developers have done, I often find myself saying, "No, that's not what I meant in my spec."
- I also often find myself saying, "Can we change this? Now that I see it, I realize this wasn't exactly what I had in mind."
- I don't need to approve screenshots or mock-ups before my developers are able to push new code to the live product.

If any or all of those situations are familiar to you, it's a good time to consider either adding a dedicated person to serve the role of product owner or shifting around the way you're approaching your role on the agile team.

What these clues have in common is that they suggest the product manager (or other person assigned to the role of product owner) is thinking of themself not as part of a cohesive agile team, but rather as someone separate from that team whose job is simply to deliver a one-way, high-level communication to the developers about what to build.

It's as though they're saying, "Here. Read this detailed spec, and get started. We'll meet again when you've built it."

One of the real advantages of the agile environment is the tight feedback cycle for both developers and product owners, particularly in sprint reviews. These will be opportunities for product owners to give developers directional feedback on what they're seeing, and for product managers to receive frequent updates on the progress of development.

FREQUENT REVIEW, COMMUNICATION AND COURSE-CORRECTIONS HELP TEAMS BUILD BETTER PRODUCTS.

It's this cadence of frequent review, communication, and course-corrections that helps teams build better products. And product needs to foster this by actively participating in story reviews, sprint planning, demos, and by creating a culture that makes your developers feel comfortable asking you questions along the way. Without this relationship, product can miss out on the real power of the agile development methodology.

All of which is to say you can't simply put in general requirements and send your developers off to work in a silo until they've built a completed product (or finished an epic or story). Product needs to be right there with them at every step.

DOES A PRODUCT OWNER NEED TO BE TECHNICAL?

A quick note on technical know-how and product ownership. Product managers can often be intimidated when asked to work closely with development, because they worry they aren't technically savvy enough to ask the right questions, or to understand the answers.

But you can't let that slow you down. When you become a true product owner and "join the agile development team," you'll need to become comfortable saying things to your developers like, "I don't understand what you mean. I'm not a developer, so you'll need to explain this to me in layperson's terms."

But there are plenty of benefits to approaching working with development this way. First, you'll often find that when your developers say no to a request of yours, it's simply because they don't understand what you're asking. (They're not product managers, after all, and aren't necessarily fluent in your language either.)

Also, when you make yourself part of the agile team you'll quickly learn a lot of the technical details that might have previously intimidated you. That's an enormous benefit both to your product's success and ultimately to your career—because it means you're learning to bridge the divide between product management and engineering that so often prevents organizations from delivering the best possible products.

TIPS FOR WORKING WITH DISTRIBUTED AGILE TEAMS

Here's a conundrum for you. The first tenet of the Agile Manifesto states that teams developing software should value "individuals and interactions over processes and tools." Two of the 12 Agile Principles make this point even more explicitly: "Business people and developers must work together daily throughout the project." And, "The most efficient and effective method of conveying information to and within a development team is face-to-face conversation."

The Agile Manifesto emphasizes the importance of people, collaboration, and face-to-face conversation, something that many agile purists will tell you means truly agile teams cannot be remote or distributed.

Yet, distributed teams are becoming increasingly popular. VersionOne's 2018 State of Agile Report found 79% of agile teams have either full or partially distributed teams. So, the good news is, while there is a clear argument to be made that real human conversations are vastly more effective than email or Slack, it is still possible to embrace the agile philosophy as a distributed team.

If your cross-functional product team is like many in today's organizations, that team spans multiple cities, maybe multiple countries, possibly even a couple of different cultures—and definitely several time zones. Even still, it is perfectly possible to be an effective remote agile team. You just need to be willing to put in the work.

How can you make a globally distributed agile product team work smoothly and efficiently? And as a member of a distributed team, how can you up your odds of success? Below, we've compiled a few suggestions for product people who find themselves working with remote team members.

> ***"Communication breakdown can happen even with teams that are located in the same office. I've seen communication get better from team members that go remote, only because they knew they needed to work at it more. They ended up being more effective in their communication after being remote."***
>
> — **Jennifer Payne**
> Quality & Efficiency Engineer and Agile Coach at ProductPlan

OVER-COMMUNICATE

When in doubt, err on the side of over-communicating. Product's role on an agile team already necessitates communicating with all of the members of the team often. And you may find that if you start driving more conversations with the team, you'll soon have a healthy culture of frequent communication.

AGILE PRINCIPLE 4

Business people and developers **must work together daily** throughout the project.

For remote teams, under-communication is one of the biggest hindrances to success, so take the extra time to communicate more, not less. And definitely do not skip out on daily standups.

DON'T RELY ONLY ON EMAIL FOR TEAM COMMUNICATIONS

You might not always be able to communicate with your globally distributed team in real-time, either by phone or video call. But when you need to communicate asynchronously—where you send a message and wait for a response—avoid the temptation to use email.

It's a safe bet that everyone on your team uses the same work email address for all sorts of business (and perhaps some personal communications). That means some of your important messages to the team could get lost, and you might not always immediately see and read their messages to you.

In addition, we have plenty of communication tools available that operate in ways that more closely imitate face to face interaction. Slack or other messaging services and video conference options are useful to fill the gap here.

FIND AN ONLINE COLLABORATION TOOL—AND CREATE A WORKSPACE JUST FOR YOUR TEAM

Instead of email, you can bring your worldwide team a little closer together by creating a special workspace just for them.

This will be your team's exclusive place to collaborate, update each other on progress, brainstorm ideas, communicate throughout the day, and maintain a complete record of all communications related to the project for later reference.

Slack channels are a great way to stay connected in real time and you can create multiple channels for different subjects to improve efficiency and reduce noise when doing focused work.

LEVERAGE ANY TIME OVERLAP YOU CAN FIND

Let's say there's just one hour of the day when both you and your overseas teammates are in the office at the same time. Maybe it's your morning and the end of the day for your developers in the Czech Republic, for example. Lock that time up—on both sides—so your team can communicate in real-time during that hour.

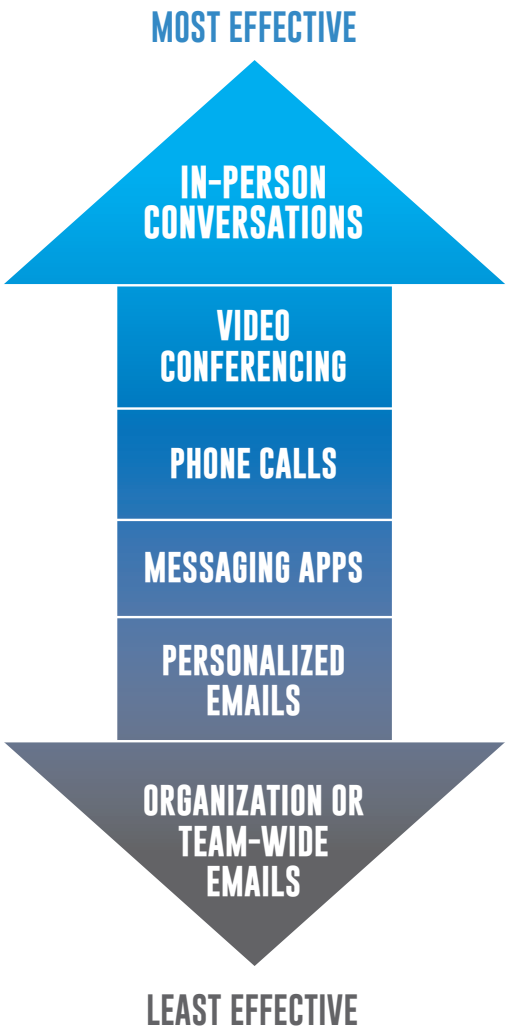
USE THE MOST RICH FORM OF COMMUNICATION POSSIBLE

> ***"I think it's universally accepted that if you want the team to perform optimally they must be co-located, but this is often impossible, so use the most rich form of communication possible given your situation.***
>
> ***The media richness theory is one that I've seen people present to help determine how to interface."***
>
> **— Nick Kim**
> Software Engineer at ProductPlan and Certified Scrum Master

If your product team is spread across several cities or even countries, you can very easily fall into the trap of viewing each other as just a series of names or email addresses—and that never leads to the magic that can come only from the chemistry of real teamwork.

So whenever possible, meet virtually through video conferencing. It'll help everyone get to know each other a little better, share their personalities with the team, and build that chemistry.

You just can't create the same sense of team cohesion over an instant-messaging platform.



Source: Media Richness Theory, Richard L. Daft and Robert H. Lengel

BE PREPARED TO WORK SOME ODD HOURS (IF YOU'RE BASED IN THE US)

When you work with teams based in other countries, you need to keep an important logistical detail in mind: While those teams might work in ultra-modern, high-tech buildings during business hours, it may be a different story when they go home.

In many countries outside of the US, the tech infrastructure might not be nearly as well-built.

Which means that for you in the United States, it might be much easier to join a call from your home during your evening or early morning than to ask your overseas team to try doing so from their homes.

MAKE EACH OTHER AWARE OF YOUR CULTURAL TENDENCIES

Funny story. The spouse of one of us here at ProductPlan worked as a product manager for a software startup years ago, and the lead developer on her team was in the US on an H-1B visa from India.

In one of their first conversations together, as the product manager walked the developer through her plans for the product, he continually shook his head. Every time she made a point, it seemed, he was signaling his disagreement. When she finally asked him about it, the developer said, "I'm not disagreeing. In my culture, the head shake is our way of saying 'Yep, got it.'" (Oh, thought the product manager. Sorry.)

From that point on, the two worked together seamlessly—which, as product manager and developer, meant they debated all the time. But each clearly understood the other.

When you run a team that spans the globe, your team almost certainly consists of members who have different cultural practices and behaviors—which can show up subtly in how they work together.

So our advice is to create a team culture of openness. Invite team members to ask each other questions if they're not sure about a message's tone, for example. Make it fun. This will not only help cut through any cross-cultural confusion, but will also help bring your team closer together.

VISIT

AGILE PRINCIPLE 6

The most efficient and effective method of conveying information to and within a development team is face-to-face conversation.

If it's possible, try to arrange for some face time once or twice a year where the entire team gets together.

DON'T FORGET THE RETROSPECTIVE

Retrospectives are an important part of every agile team's process, and remote teams aren't exempt from this. Distributed teams need to talk about communication and availability during every retrospective session they have.

> ***"Make it clear that it's ok to say, "So and so wasn't available for me to get my questions answered." It isn't to say that someone isn't doing their job, it could be that they have taken on too much and it's time to hire someone else. Because communication with remote teams is more challenging by nature, retroing on how the process and communication is going is a must!"***
>
> **— Jennifer Payne**
> Quality & Efficiency Engineer and Agile Coach at ProductPlan

AND REMEMBER:

AGILE VALUE 1

Individuals and interactions over processes and tools.


3
AGILE PRODUCT MANAGEMENT
IN MOTION

3 AGILE PRODUCT MANAGEMENT IN MOTION

BALANCING STRATEGY AND TACTICS

To some product managers, the phrases "Agile" and "Product Roadmap" might seem to be contradictions in terms. After all, a roadmap suggests strategic planning and a long-term product vision. Agile development, by contrast, suggests short-term cycles and frequently shifting priorities.

Does it even make sense to attempt to have a long-term strategy when you're working under a philosophy that carries so much uncertainty?

Of course it does. Embracing the agile philosophy and incorporating agile principles into product strategy can help product teams make more intelligent decisions, build more successful products, and innovate more rapidly than ever before. A few key benefits of applying agile principles to your product strategy include:

- Validation and testing happen early and often as agile principles prioritize shortening time to market and iterative development.
- Agile product strategies put the customer front and center and enable teams to quickly adapt to changing market needs and customer feedback.
- Daily communication between business and development staff encourages constant alignment between business and product objectives and helps support transparency.

Now that you know more about the why, let's discuss the how. This chapter is all about empowering product managers to successfully merge strategic plans and long-term product vision with agile development principles.

First, here's a high-level overview of what the relationship between strategy and tactics in agile product development often looks like.

STRATEGY	
Inputs: ■ Business Objectives ■ Stakeholders ■ Market Feedback & Customer Feedback ■ Innovative solutions to unmet needs	**Outputs:** ■ Product Vision ■ Product Roadmap

Purpose:

- A mid to long term strategy serves as a north star to guide tactical decisions and maintain focus on producing long term results.
- The product vision and roadmap are used as communication tools to get executive buy in and align the organization on a unified front.

TATICS	
Inputs: ■ Product Vision ■ Product Roadmap	**Outputs:** ■ Product Backlog ■ Release Plan ■ Tasks: User Stories, Tasks, Bugs etc.

Purpose:

- The backlog gives the development team a prioritized list of tasks to tackle during sprints.
- Breaking down broader strategic goals into smaller pieces to deploy piece by piece ensures frequent opportunities to gather feedback and correct course.

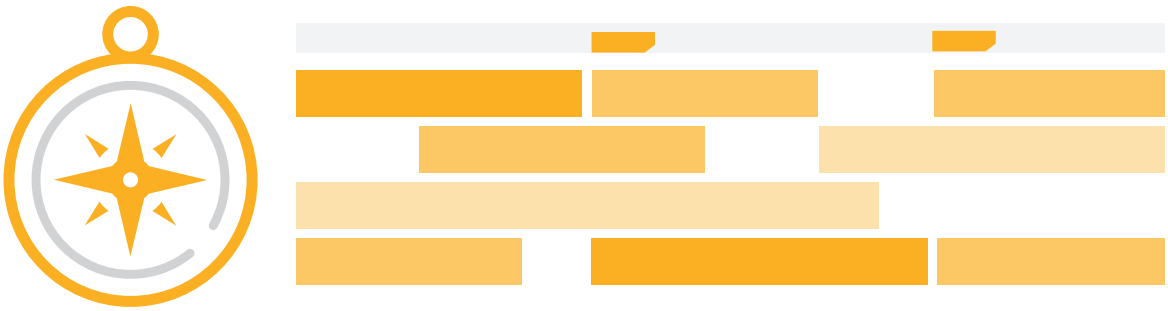
THE PRODUCT ROADMAP AND THE PRODUCT BACKLOG

Now, let's take a look at the two important tools product people need to work within a balanced agile environment: the product roadmap and the product backlog.

The product roadmap and product backlog serve distinct but complementary roles in helping a product manager shepherd a product from early strategic conceptualization all the way through development and market release. An agile product roadmap speaks in terms of epics, themes, outcomes, and goals, while the backlog is the detailed features and other tasks that deliver the product. In a sense, the backlog is a translation of how your team will deliver the vision outlined on an agile product roadmap.

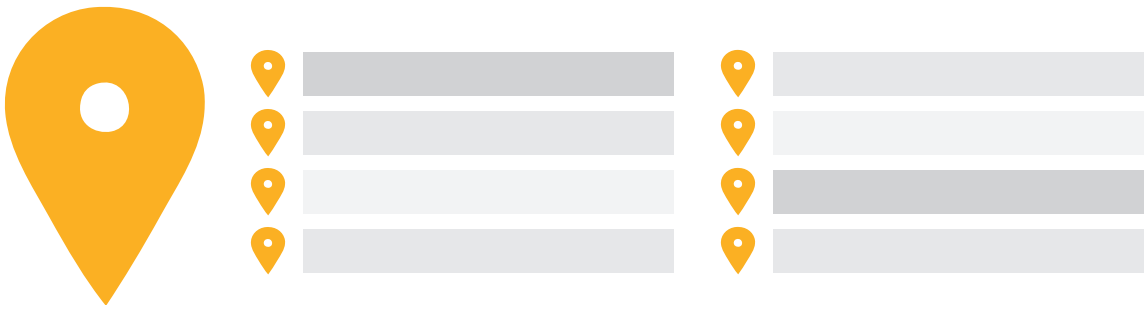
	PRODUCT ROADMAP	PRODUCT BACKLOG
Content	High-level: themes and epics or outcomes and goals	Task-level: user stories and defects
Audience	Executive team (and other stakeholders)	Primarily for product and development teams
Intent	Conveys a strategy	Conveys tactical steps in execution of plan
Time Frame	Varies, typically ~3 months	1 or 2 sprints

PRODUCT ROADMAP



The product roadmap is a high-level tool to help a product manager capture, organize, and communicate their product's strategic goals and plans. A quick glance at your roadmap should convey where you are taking your product over the coming months (and even years), as well as your strategic reasoning for doing so—whether it's because of an opportunity you've identified in the market, strong user demand, competitive pressure, or a combination of factors. Typical items on a roadmap will include high-level product themes and possibly epics.

PRODUCT BACKLOG



The product backlog lists and prioritizes the task-level details required to execute on the strategic plan set forth in the roadmap. A quick glance at your backlog should convey what's next on your development team's to-do list as they execute on your roadmap's big-picture vision. Typical items on a product backlog include product stories, bug fixes, and other tasks.

WHY THE PRODUCT ROADMAP AND PRODUCT BACKLOG MUST BE SEPARATE

As you can see, the product roadmap and product backlog both serve important functions in a product's development, but these functions are very different. This is why these tools should be used in conjunction with each other—but not interchangeably.

Without a high-level overview of your product's strategic objectives and plans, you cannot effectively build a useful list of tasks, in priority order, for developing the product. So you need a standalone, clutter-free, strategic roadmap to capture and communicate this strategy.

At the same time, until you and your team translate your roadmap's big-picture ideas and strategies into an actionable list of specific tasks—in other words, the backlog—your roadmap won't do much good in helping you drive your product's actual development. So, equally important to your roadmap is an organized and intelligently prioritized product backlog to help keep your development team focused on the right tasks at the right time.

This is also why you do not want to mix these two tools or use only one to serve both purposes.

A ROADMAP WITH TOO MUCH TACTICAL DETAIL CAN CAUSE TEAMS TO LOSE SIGHT OF THE STRATEGIC BIG PICTURE, AND A BACKLOG THAT FOCUSES ON HIGHER-LEVEL, STRATEGIC ITEMS CAN LEAVE THE TEAM WITHOUT A CLEAR PLAN FOR WHAT GROUND-LEVEL TASKS TO TACKLE NEXT.

HOW TO USE THE PRODUCT BACKLOG IN CONJUNCTION WITH THE PRODUCT ROADMAP

If the roadmap and backlog should be used together, as complementary tools, how should a product team go about this? Here are a few suggestions.

THE PRODUCT ROADMAP COMES FIRST

Before you can begin prioritizing specific product development tasks, you first need to have an overarching strategic vision for the product—your "why." You will almost certainly also need to receive the green light from your executive team and other stakeholders to move forward with your product's strategic plan before you can begin deploying budget, developers' time, and other company resources to executing on that plan.

For these reasons, you should develop your roadmap first—and determine your product's high-level objectives, priorities, and plans. Then you can begin translating that big-picture strategy into a backlog of tasks, stories, and specific action items for your development team.

ALWAYS KEEP YOUR ROADMAP AND BACKLOG IN SYNC

It will also be important that you make sure your roadmap and backlog are actually working together—which means they're always in sync, and are both up-to-date, as opposed to becoming increasingly standalone documents that tell different stories.

In practical terms, this means regularly referring back to your roadmap to make sure the items prioritized on your backlog still represent the roadmap's strategic objectives and priorities. It also means always keeping your backlog organized and lean, and never allowing it to become a black hole where everyone simply throws in their ideas and requests.

THE STRATEGY: AGILE PRODUCT ROADMAP

A plan without a strategy is like a ship without a captain. And that's why the product roadmap always comes first.

An agile product roadmap communicates to the organization the big picture – the initiatives that move the needle, expand markets, address competition, and create customer value. That big-picture thinking can't be distilled in the backlog. It's challenging to communicate strategy in a list that's 200 items long, especially to executives and other stakeholders who may not think in terms of iterations or sprints.

It's important for product managers not to make the mistake of thinking that because they have a roadmap, they're not agile. Those two concepts actually work in tandem. You need a roadmap to set the strategic goals for your company, but you still have a lot of freedom to move things around within those goals.

WHAT MAKES A PRODUCT ROADMAP AGILE?

An agile roadmap is a living document rather than a plan set in stone. Like the backlog, the product roadmap should be regularly discussed, prioritized, estimated, updated, and shared.

Because an agile product roadmap will inevitably change, it's important to set expectations with your stakeholders that the roadmap is not a promise. Keep the roadmap dates at a monthly or quarterly level, or leave the dates off altogether to avoid setting the impression that features will be delivered by a specific date.

Product owners need to regularly communicate where the product is heading so that everyone is on the same page, especially to stakeholders who make final decisions, control the budget, or influence the direction of the company. Your

agile product roadmap therefore should be a visual, easy-to-digest document that your stakeholders can understand and that gives perspective to your backlog.

BEST PRACTICES FOR AGILE ROADMAPS

- Frequently adjusted to incorporate change and new findings.
- Expectations are set with stakeholders that things can and will change on the roadmap.
- Features high-level themes and a strategic view of mid to long term direction rather than detailed feature lists and release dates. (offers guidance rather than a strict project plan)
- Different versions are available for communicating the strategy with different audiences.

TRANSLATING STRATEGIC INITIATIVES TO WORKABLE TASKS

One important aspect of a successful agile product management process is bridging the gap between the strategic and the tactical. Once you've completed the first strategic stages of product planning, you somehow need to translate the plan delineated on your roadmap into actionable steps for your engineering team to complete.

One common agile practice is slicing the fully-formed features on your roadmap down into user stories.

A **user story** is a small, self-contained unit of development work designed to accomplish a specific goal within the product. It describes the intent or desired outcome of a small but complete slice of user functionality. A given product feature may be comprised of several user stories.

A story is usually written from the user's perspective and follows the format:

> ***"As [a user persona], I want [to perform this action] so that [I can accomplish this goal]."***

As you can see, a user story doesn't usually describe exactly how a feature is to be implemented, but rather what a user is hoping to achieve and why. The implementation details are usually included or attached as a part of the stories.

For example, a user story for our roadmap software might be written this way:

> ***"As a product manager, I want to search across the Notes in my company's product roadmaps so I can quickly find initiatives that require my team's resources."***

WHY USER STORIES?

STORIES ARE VERTICAL: THEY ALLOW THE DEVELOPMENT TEAM TO FOCUS THEIR EFFORTS ON A COMPLETE PIECE OF FUNCTIONALITY (NO MATTER HOW SMALL)

One reason for favoring user stories is that they are small vertical slices—meaning they represent an entire piece of functionality. In other words, a story should allow a user to complete some task, even if that task is small. These tasks represent verifiable incremental steps toward your larger goal or milestone.

AGILE PRICIPLE 1

Our highest priority is to satisfy the customer through early and continuous delivery of valuable software.

STORIES CAN EASILY FIT INTO SPRINTS, WHILE FEATURES GENERALLY CAN'T

Another benefit of breaking your product's desired functionality into user stories is that, because they represent very small chunks of the product, stories make for more effective scheduling and collaboration.

If priorities change and you need to adjust your development schedule, it will be much easier to shift stories around on the agenda than it would be to ask your development team to halt progress on a large feature that's 65% complete—and refocus on an entirely new, large feature.

AGILE PRICIPLE 3

Deliver working software frequently, from a couple of weeks to a couple of months, with a preference to the shorter timescale.

STORIES CAPTURE INTENT AND DESIRED OUTCOMES, LEAVING THE ACTUAL DEVELOPMENT INSTRUCTIONS TO THE EXPERTS: THE DEVELOPERS THEMSELVES

Stories give the product manager the opportunity to describe, very specifically, what the desired outcome of a piece of functionality will be for a user—such that any skilled developer will be able to understand the intent, perhaps with a few clarifying questions. Properly worded stories give your developers the breathing room they need to do their best work.

AGILE PRICIPLE 5

Build projects around motivated individuals.
Give them the environment and support they need, and trust them to get the job done.

4 TIPS FOR DEVELOPING EFFECTIVE USER STORIES

1. MAKE IT A TEAM EFFORT

Product management really has to be a true team effort. And that can't simply be a matter of your saying that it's a team effort. You have to mean it, practice it, and build it into your routine. That means optimizing your communication processes. It means being available to quickly respond to questions.

It also means not falling into the trap of having just one or two people on the cross-functional team—the product manager and the engineering lead, for example—do all of the brainstorming and breaking down the stories into fully formed plans and then just presenting those to the rest of the team. Your stories in many cases really should arise out of discussions with your larger cross-functional team.

2. USE THE RIGHT TOOLS

Optimizing your team's ability to collaborate—whether in a large brainstorming session, during a quick stand-up, or even through a quick back-and-forth across time zones—requires the right tools.

This starts at the very beginning of your product strategy planning—with the right product roadmap application. And it extends to your cross-functional team's ability to communicate quickly and effectively every day, no matter where they are—by using the right project collaboration software.

And ideally, it means finding tools like these that can talk to each other—so your team can check their daily progress against their big-picture goals, for example, and sync their roadmap's progress with the current status of each individual task.

3. BREAK STORIES DOWN AS SMALL AS YOU CAN

For your stories to be as effective as possible—clear to your developers, simple to complete, easy to test and gain internal acceptance on—you want to break them down into the smallest logical units that you can.

Image you are writing user stories for a new admin console. While your initial inclination might be to write something like "Admin manages users", you would probably be better breaking that story down into three smaller stories, such as "Admin adds a user", "Admin edits a user", and "Admin deletes a user".

Now your developers have a set of clear, highly specific tasks they know how to complete. And, as a bonus, you've broken down your complex story into enough component pieces that you could even afford to de-prioritize one if needed. For example, you could ship an early version of your product that allowed administrators to add and edit users, but not delete them.

> *"Because big stories have a lot of complexity, they have a lot of risk as well. What if you get 80% or 90% down the path and realize the story was based on erroneous assumptions? Break big stories into smaller stories—representing at most a few days of development."*
>
> **— Dan Podsedly**
> VP of Pivotal Labs

4. DEVELOP A STRATEGY FOR COLLABORATIVE STORY ACCEPTANCE

Finally, one often-overlooked strategy is to bring more people into your story review process. When your developers check in a story as completed and you simply hand it off to QA, you miss a lot of opportunities both to catch issues with the story as coded and to distribute product and user-persona knowledge across your broader team.

It can be helpful to develop a plan for achieving "collaborative acceptance" for your user stories. What this means is bringing in your designers, your developers, the rest of your product team, and of course your testers, to review your stories. But the distinction here is that these groups aren't going to be looking only for bugs in the code or typos in the UI.

Your broader team is going to bring their respective areas of expertise and understanding about your user personas to these reviews—and offer feedback from all angles, not just bug fixes.

The additional bonus of this strategy is that you will be growing a broader and deeper knowledge base in your company about your product and its users—which will ultimately lead to better, more effective user stories coming from all across your team.

THE PLAN: PRODUCT BACKLOG

In most agile product development organizations, the backlog is used by the development team to track what's coming next, at least for the next few sprints or iterations. Many agile teams rely heavily on the backlog, as it maps out short-term initiatives.

In an ideal situation, your product backlog will be a lean, organized, and prioritized list of the things your team needs to do next. It could include product features, stories, tasks, bug fixes, and any other product-related items that need handling. In other words, your backlog should function as a well-ordered to-do list of important product action items.

YOUR BACKLOG SHOULD FUNCTION AS A WELL-ORDERED TO-DO LIST OF IMPORTANT PRODUCT ACTION ITEMS.

> ***"A healthy backlog is one where near term high priority stories (1-2 sprints out) are groomed well, and the further you get down the list the larger the stories are."***
>
> — **Nick Kim**
> Software Engineer at ProductPlan and Certified Scrum Master

But the key—remember, we're still talking ideal situation here—is that every item on the backlog is clear and easy to understand. Items should earn their place on the list, be slated for near-term action, and be prioritized appropriately in relation to other items. And just as important, nothing should be on the backlog that doesn't belong there.

But if your backlog is like most, it probably looks more like a free-for-all list—a long, ever-expanding and unorganized document capturing every thought anyone on your team has ever had about your product. Good ideas. Lousy ideas. Low-priority tasks. High-priority tasks. Super-burning-hot crisis-management fixes. Brainstorms for other products. A grocery list? (Well, that was probably a copy-paste accident.)

And because the list becomes so long and so difficult even to look at, you lose track of which items are high priority. You have a more difficult time finding the right items to place into your next sprints or longer-term development cycles.

In fact, unlike the ideal scenario in which every backlog item is all but guaranteed to see some near-term action, the items on your backlog are just as likely to disappear and never be heard from again.

We are passionate about helping product managers stay organized and able to focus on their strategic vision. And other than poorly executed product roadmaps, we've found that ineffective backlogs are often the biggest hindrance to a product person's ability to successfully drive a product forward. Here's some suggestions for managing your backlog and ensuring that it does not become a black hole.

BEST PRACTICES FOR MANAGING AGILE PRODUCT BACKLOGS

1. ARRANGE THE TOP ITEMS ON YOUR BACKLOG TO REPRESENT YOUR NEXT SPRINT

One helpful step to organize your product backlog is to arrange the top portion of the list as the contents of your next two sprints.

This way you aren't constantly looking at the backlog and asking, "When will we get to this?" and "When can we start tackling that?"

Using this strategy, the top items on your backlog aren't just "top priority" tasks with no internal dates associated with them—they also have a built-in timeline: your next sprint.

Of course, you'll need a mechanism for determining what items should be included in your team's next sprint, and we'll discuss ideas for that below. (In the next section we'll talk about where backlog grooming may land in your agile process.)

2. DON'T INCLUDE ANY TASK LOWER THAN SECOND-LEVEL PRIORITY ON THE BACKLOG

This is another simple, clean way of determining what makes it onto your backlog and what needs to go somewhere else (like a "Longer-term Tasks" file). Priority level two is a logical cutoff point for what makes it onto your backlog, and here's why.

You've been in brainstorming meetings where the team jots down 20 viable product ideas on the whiteboard. Maybe you've even hosted these meetings. Obviously, you can't execute on all 20 of those ideas, at least not in any near-term timeframe. So what do you do? You prioritize: Maybe you select the best two or four of those ideas and break them into stories, tasks, and plans your team can start working on.

As for everything else on that whiteboard, you'll capture it, of course, but you can't put it all on your backlog (or, even more unrealistically, on your roadmap). The product backlog needs to remain as lean and realistic as possible. It should contain the things on deck for your next sprint, and the second-level priority items you'll get to within the next few months. And that's it.

3. CREATE A SEPARATE LIST FOR LOWER-PRIORITY (OR LONGER-TERM) IDEAS AND REQUESTS

As a product manager, you're certainly no stranger to great ideas for your product. You have them. Your customers have them. Your marketing or sales folks have them. But not every good idea can be implemented right away — so you do need a place to capture and store ideas that are worthy but need to be shelved for the time being.

And for two very important reasons, you can't afford for that place to be the backlog.

Product managers who simply toss every request, idea, and task onto the bottom of their product backlog—because they have no other trusted place to capture and store those items—make every future review and reassessment of their backlog more difficult. They also make it more likely that they will miss something important when they look over their backlog.

PRODUCT MANAGERS WHO SIMPLY TOSS EVERY REQUEST ONTO THE BOTTOM OF THEIR PRODUCT BACKLOG MAKE FUTURE REVIEW AND REASSESSMENT OF THEIR BACKLOG MORE DIFFICULT.

What's great about creating a separate list for less-urgent product-related items is that it helps you keep your product backlog limited to those tasks that are truly urgent or of high strategic value. Which means it keeps your product backlog itself more strategically valuable.

So create other lists to capture your product-related ideas that don't earn a spot on the backlog—such as a "Great Ideas" file, and maybe a "Longer-Term Tasks" list.

4. ASSIGN SCORES (OR USE SOME OTHER QUANTIFIABLE SYSTEM) FOR DETERMINING EACH ITEM'S OVERALL VALUE

At ProductPlan, we've included a weighted scoring tool in our product roadmap app. We've found that when dealing with a finite amount of time, budget, and development resources, product managers need a mechanism to quantify (or "score") the overall strategic value of each proposed feature or task against all of the others—to determine which will give their product the biggest strategic advantage.

But you can, and should, take a similar system to scoring the benefits and costs of items on your product backlog.

We recommend using a scoring model—whether based on ProductPlan's suggested metrics including "Customer Value," Increased Revenue" and "Implementation Costs," or using some other system—to score each item competing for a slot on your backlog.

Some items will earn a spot in your short priority one list (planned for work in the next sprint), others will make it to priority level two (planned for development in, say, the next three months), and everything else will find itself in your "Longer-Term Tasks" file. But when you've organized your list this way, you'll know exactly why every item is where it is on your list, and you'll be able to explain and defend your strategic thinking to your stakeholders and other teams.

5. FIGURE OUT A POINT SYSTEM FOR ASSIGNING TIME AND DEVELOPMENT RESOURCES TO EACH ITEM

When prioritizing your backlog, one important factor to keep in mind for every task is how much work it contains. As a product person, you are unlikely to be held responsible for coming up with estimates. Generally this task is up to the Scrum Master or project manager and developers to tackle. But you should still understand how estimation works.

Some teams estimate work by number of hours each task will take, but many vouch for "story points" instead. The reason being, not all developers work at the same pace. So a task which may take one developer 4 hours may take another 6 hours.

STORY POINTS PROVIDE A STANDARDIZED ESTIMATE THAT EACH DEVELOPER CAN INTERPRET RELATIVE TO HOW THEY WORK.

It's important to remember that your team is unique and has a unique set of skills, strengths, and weaknesses. This is why the backlog can play such an important role in your product and development teams' planning sessions. If you know you

have only one or two developers who have the skillset or experience to handle a certain story or feature, you need to budget the time (the "points") of those developers carefully as you assign other tasks for your upcoming sprint.

A word of caution for product people about estimates...

> ***"Estimates are just that, estimations. Once the team dives into the work, they may find they took on too much because of new information they discovered after beginning the work. It is important that the team understands they can re-adjust their commitments and SHOULD communicate with product if the expected work to be completed has changed. This is one of the biggest causes of stress I've seen in teams."***
>
> — **Jennifer Payne**
> Quality & Efficiency Engineer and Agile Coach at ProductPlan

6. RE-EVALUATE THE LEVEL ONE AND TWO ITEMS ON YOUR BACKLOG REGULARLY

Finally, it's important to keep in mind that your product backlog is a living document—changing in priority often. The top portion of your backlog should be disappearing after every sprint, as your team completes it. This means that some portion of the second-level items on the backlog will be moving up after every sprint as well, to the on-deck spot.

When you've followed the other suggestions we've offered here, and every item on your backlog now has a strategic reason for being exactly where it is on the list, you'll find it much easier to review that list regularly to determine if any new information—competitive intelligence, customer requests, or just a screaming-hot urgent fix—demands you re-prioritize things.

Most importantly, you won't be guessing. You'll be making those prioritization decisions systematically.

7. STANDARDIZE YOUR BACKLOG

If your organization is like most, just about anyone can add an item to your backlog. When things are manageable, this process is fine.

But what if a QA tester adds an item to your backlog that reads something like this: "Sporadic blockage of internal transmission messages detected."

"Huh?" you wonder as you read this new backlog entry. What in the world does that mean? Is it important? Will it affect the customer experience?

Now, if your backlog is manageable—only a few items, all well-organized—you'll catch this little outlier quickly and be able to deal with it. (Starting by asking the QA person: "Huh?") Because you keep your backlog lean, a potentially serious item like this won't sit unattended forever.

But if the backlog is already way too long, adding yet another item all but guarantees the new item will be buried for a good amount of time before you catch it. And even then you won't be sure whether or not it's a priority, because you won't know what it means.

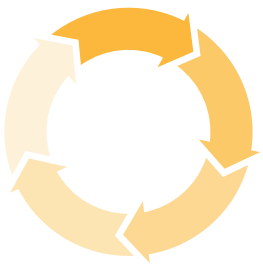
Set a new rule for anyone placing items in your backlog. Demand a plain-language explanation of the issue, feature, or defect. No developer-speak. No QA-speak. No court-reporter-style shorthand notation. Every item included on your backlog must be written in a way that is both understandable and actionable by the product owner.

And don't forget: This rule should apply to you as well. Why? Because if your backlog is a black hole, that almost certainly means there are items on it that are many months or even several years old. If you aren't clear in your description of an item you include on the list, what are the odds that you'll remember what you meant 18 months from now?

AGILE DEVELOPMENT CYCLES

At the risk of sounding like a broken record, there is no single "right" way to put agile principles in practice. Agile principles are typically adapted into processes that meet the unique needs of each team they serve. But, to demonstrate how cross-functional teams may elect to work together during their rapid development cycles (sometimes called sprints), let's outline the general flow that many agile teams follow for development cycles and share a few pointers for getting the most out of the philosophy. What we've outlined reflects a flow that some but not all agile product development teams follow.

	TACTICS:	WHO:
STRATEGIC PLANNING	■ Product Vision ■ Product Roadmap	■ Product Manager and Key Stakeholders
TACTICAL PLANNING	■ Backlog Grooming ■ Sprint Planning	■ Product Manager, Product Owner, and Agile Team
BUILD	■ Pair Programming ■ Daily Standups ■ QA	■ Delivery and Product Teams
SHIP	■ Sprint Demos ■ Deployment	■ Delivery and Product Teams
LEARN	■ Customer Feedback via support, analytics, & customer interviews ■ Retrospectives to review team's process	■ Product Manager and Product Owner ■ Agile Team



REPEAT

STRATEGY

As we discussed, strategy comes first. Only once that is established and agreed upon by all relevant stakeholders can the rest of the process begin.

BACKLOG GROOMING AND PRIORITIZATION

Agile embraces change, and as such, agile processes need to facilitate that too. The backlog is not a static document. You need to refine, re-prioritize, and update the backlog frequently. Regular backlog grooming ensures product plans effectively serve changing business and reflect market changes and customer feedback.

AGILE PRICIPLE 2

Welcome changing requirements, even late in development. Agile processes harness change for the customer's competitive advantage.

How often should agile product development teams groom their backlog? Our 2019 Product Planning Report found the vast majority of product teams re-prioritize their product backlog weekly or monthly.

40% OF PRODUCT TEAMS GROOM THEIR BACKLOG WEEKLY.

Source: 2019 Product Planning Report

Backlog grooming is often a project owned by the product team, however it is wise to get other members of the cross-functional team involved too. For example, you can work with QA to further flesh out stories in the backlog ahead of a sprint planning meeting. QA can help you determine whether stories in the backlog contain enough detail and context. They can help tell you whether stories are easy for developers to understand, and even suggest when to break down larger stories into smaller, more digestible ones.

BACKLOG ESTIMATION

Many agile development teams use estimates to predict how much work stories in the backlog contain. While product is typically not heavily involved in coming up with these estimates (they require deep technical insight into development concepts), they are sometimes involved with estimation. The important thing to remember is estimates are not concrete but fluid, and therefore you may need to shuffle things around even mid-sprint if the team uncovers more complexity than anticipated.

SPRINT PLANNING MEETINGS

Before a sprint begins, you need to get the entire team together to set goals and expectations. You can use sprint planning or "kickoff" meetings for this, where everyone gathers to decide which stories to commit to.

You want to spend this time together as efficiently as possible. That means as a product person, you need to come prepared. Your developers will typically ask questions about the user stories in the backlog and it is your responsibility to ensure they understand the vision, requirements, and often the higher-level strategic goals each story seeks to serve.

This is a critical moment for the product manager because it's when they need to provide the development team with all the resources and information that they will need to successfully develop and execute during the sprint.

It's important to remember that it's up to the Scrum or project manager alongside the development team to make decisions about how much work is appropriate to commit to during the sprint timeline. It's not in your best interest as a product person to pressure the team into biting off more than they can chew.

AGILE PRICIPLE 8

Agile processes promote sustainable development. The sponsors, developers, and users should be able to maintain a constant pace indefinitely.

Furthermore, because these commitments are based on estimates (which can be a slippery slope) of how much time they'll require, it's advised that you accept that the estimates (and therefore commitments) are typically fluid.

At the end of a sprint planning meeting you should have agreed to a goal and the specific items within the backlog that will be complete by the end, and your development team should have a deep understanding of each story they've committed to.

DAILY STANDUPS

Good software isn't developed in silos. You need to communicate daily with the team.

AGILE PRICIPLE 4

Business people and developers **must work together daily** throughout the project.

Many agile product development teams put this in practice through daily standups, quick sessions where each member of the team shares what they accomplished yesterday, what they'll try to accomplish today, and where blockers are. There is some debate as to whether product people should attend daily standups, but we believe that more communication is always favorable over less.

Plus, since these round robin type meetings are typically no longer than 15 minutes, it's a small price to pay for more insight into progress. And your presence during the daily standups gives developers yet another opportunity to ask you for clarification or support on the stories they're working on.

DEMOS AND SPRINT REVIEWS

At the end of a sprint, many agile teams will have formal demos to share the work they've completed during the sprint. Sometimes demos include a small group of selected stakeholders, and other times the entire team is invited.

AGILE PRICIPLE 7

Working software is the primary measure of progress.

One benefit of inviting a broader audience to attend sprint reviews is it presents a great opportunity for product people to get feedback.

Usually developers will only present completed work during these reviews. As a product person, you want to make sure you're seeing demos of stories as early as you possibly can. So do your best to get demos of work-in-progress even during the sprint.

Note that some developers will be uncomfortable sharing "imperfect" software with you, so it's up to you to build a culture of trust within your cross-functional team. Build relationships with your developers and be sure to reiterate that you are not asking to see demos to micromanage, but because you want to be able to share feedback and guidance as early and often as possible. If you're smart about it, you'll be able to foster a culture where developers come to you to share work in progress and get your feedback.

RETROSPECTIVES

Nothing in agile is set in stone. Your team will change, roles will change, and processes will change too. At the end of the sprint or development cycle it's time to reflect on how things went. Agile principles encourage teams and individuals to continuously assess themselves and adapt to work better together.

This is where the all-important sprint retrospective comes in.

AGILE PRICIPLE 12

At regular intervals, the team **reflects on how to become more effective**, then tunes and adjusts its behavior accordingly.

Regularly occurring product retrospective meetings can help your team find strengths and weaknesses in your current processes. They help you identify, analyze, and solve problems, with actionable plans to prevent the same hiccups from happening again.

If done well, a product retrospective presents an excellent opportunity for your team to get together and critically discuss recent cycles while celebrating successes and learning from mistakes. Ideally, at the end of a productive retrospective, everyone leaves the room feeling heard, understood, and optimistic.

But mastering the art of a productive retrospective meeting takes some time and experience.

TIPS FOR EFFECTIVE RETROSPECTIVES

CREATE A SAFE SPACE

A product retrospective meeting can be a lot like a therapy session, so it is important to create an environment that feels safe and secure. If people are afraid to voice their concerns and frustrations, or otherwise be honest about their thoughts, it will be difficult to get the full benefit of a retrospective meeting. You need to break down any walls of fear that might prevent your team from feeling safe, because you cannot prevent problems from repeating in the future if you don't talk about them now. Leadership needs to be supportive of this by encouraging open communication and meaningful conversation from everyone included. The last thing you want is to make your team feel afraid they will get in trouble for admitting to mistakes or speaking honestly.

Who you invite to your retrospective can play a significant role in your team's comfort level. It's wise to keep your attendee list as minimal as possible. You should also be wary of inviting executives, managers, or any other people whose presence might make the team clam up.

MAKE RETROSPECTIVES A HABIT

Establish a cadence for your retrospectives and, most importantly, stick to it. Even if you and your team don't feel like there is much to discuss, you'll be surprised what can be discovered once everyone is in a room together.

Thinking retrospectively outside the confines of a retrospective meeting is a habit you should instill in your team. While product retrospective meetings are specifically designed forums for this type of thinking, you should encourage your team to be constantly retrospecting. You can help support this by adding a few minutes on to your daily standups or weekly team meetings for a "micro retrospective" where everyone shares something that went well and something that went not so well with the team. Micro retrospectives can't possibly replace retrospective meetings, but they are a good way to keep people thinking in the interim.

DON'T FORGET TO LOOK AT THE POSITIVES

While trying to find ways to improve, it can be easy to forget to mention the things that your team already does well. This is particularly significant if your team is composed of perfectionists who have high expectations for themselves and others.

Make sure you allocate some time to discuss the small wins. These are also opportunities to learn! If you focus only on the negative aspects of your recent initiatives, you risk making your team dread retrospectives. Retrospectives aren't meant to be platforms for beating yourself up. Rather, they're meant for reflection, so remember to reflect on all aspects of recent work, not just the things that went less than perfectly.

RETROSPECTIVES AREN'T PLATFORMS FOR BEATING YOURSELF UP. RATHER, THEY'RE INTENDED FOR REFLECTION.

THINK OUTSIDE THE OFFICE

Remember NIHITO (nothing important happens in the office)? Whether or not the phrase rings true, you can also apply the concept to your retrospectives by taking them elsewhere. A new environment can be beneficial in two ways. First, new surroundings can help fuel creativity and promote deeper thinking. Second, switching up the scenery can make the whole process more fun. It never hurts to have your entire team excitedly looking forward to the next product retrospective. Consider experimenting with different locales: is there an outdoor area nearby where you can host your retrospective? What about a new brunch spot in town? Your change of scenery could even be as simple as renting out a meeting room of a nearby co-working space.

Another way you can switch things up is by experimenting with different formats and structures for your retrospectives. Consider breaking the team up into pairs and having each pair facilitate a different part of the discussion. Or have each pair host their own product retrospective and share their notes with the team as a whole. Remember, there is no one size fits all approach to retrospectives. What matters most is that you're able to find a process and format that delivers positive results for the entire team.

BRING IN BACKUP

You need a designated person to run your product retrospective meetings. This person is responsible for watching and "reading" the room, guiding conversations in a productive direction, and ensuring you're staying within the set agenda. Many teams find it most comfortable to invite someone outside of the team such as an agile coach or other objective party with experience running these types of meetings as a designated facilitator. But, if someone on your team is willing and able to facilitate, that is perfectly fine as well, as long as you make it clear what their role in the discussion is before the meeting starts.

SOMETIMES YOUR PRODUCT RETROSPECTIVE MEETINGS NEED RETROSPECTIVES OF THEIR OWN

Remember, you can't fix the problems you do not discuss. This applies to those related to your retrospective meetings as well. You don't need to have a formal retrospective for your retrospectives, but it's good to allocate a small chunk of time at the end of your retrospective meeting to make sure that the way you're doing things is useful for everyone involved. Retrospectives are for everyone's benefit, so if half of the team isn't getting much from them, you need to make adjustments for the next time around. After all, part of creating a safe space is ensuring everyone is happy to be there.

DON'T SKIP THE RETROSPECTIVE

Investing some time and effort into making your product retrospective meetings as effective and productive as possible will help your entire team get the most out of this unique learning opportunity. There are plenty of general recommendations you can consider when trying to improve your retrospectives, but at the end of the day it's up to you and your team to identify what works best for you.

ACCOUNTABILITY MATTERS

Remember that while the act of retrospecting in itself can be a cathartic process for your team, the end goal is to make improvements. You need to establish accountability for making these changes by following up at regular intervals.

> ***"We often identify action items in retrospectives, and never come back and revisit them to make sure they were handled. I think this is crucial, otherwise these become gripe sessions and aren't very productive."***
>
> — **Nick Kim**
> Software Engineer at ProductPlan and Certified Scrum Master


4
EVANGELIZING THE AGILE MINDSET

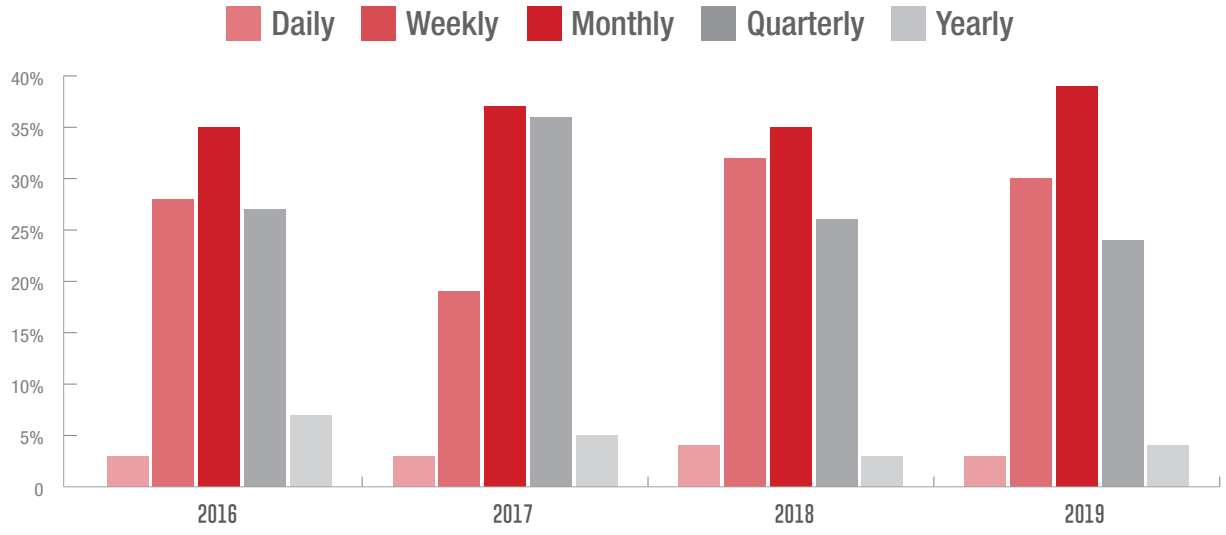
4 EVANGELIZING THE AGILE MINDSET

WHY ISN'T EVERYONE AGILE?

Statistically speaking, the software development world is going agile. Over the past years we've seen a continuing trend and an increasing shift from waterfall to agile.

In 2016 ProductPlan ran its first annual Product Planning Report in which we surveyed product managers and roadmap owners around the world. At the time, 62% of product managers reported updating their product roadmaps weekly or monthly. Then, our Product Planning Report found the number of companies updating their roadmaps once a year dropped sharply between 2016 and 2017—by almost half, in fact. In recent reports, the trend toward more frequent roadmap updates has continued.

HOW OFTEN DO PRODUCT TEAMS UPDATE THEIR ROADMAP?



Source: 2019 Product Planning Report

While the vast majority of software development organizations today report they are "agile," there are many organizations in the software development field still stuck in their waterfall ways—some of them don't even realize they are.

Transitioning from traditional software development philosophies to embrace the agile way of thinking is not something that happens overnight. As we've already discussed, following a prescriptive framework for agile development does not make you an agile team. Agile is a way of thinking, and as such you can't flip a switch and suddenly be agile, it's an ongoing process.

Perhaps this explains why there are many organizations out there who claim (and maybe even believe) they're "agile" but are in fact, still very much rooted in tradition. Often, these teams want to work by agile principles and see the value of them but are yet to fully harness the benefits of the agile mindset.

For example, it's not uncommon for development teams to attempt to make the transition to agile by implementing aspects of agile processes (such as daily standup meetings) only to continue on with waterfall mentalities. So from a process standpoint the team may appear agile, but in practice they may have long development cycles where comprehensive documentation is required up front. For these organizations, evangelizing change may be as easy as incrementally making changes over time.

But then there's the other group of organizations not working with agile principles. These are the organizations that continue with a waterfall approach because that's how their staff and executives have always done it. When attempting to evangelize agile in these organizations you may face resistance in the form of mentalities like "if it's not broken, don't fix it," and "we're not changing our process because we've always done it this way."

AGILE IS A TEAM SPORT

You really can't make agile product management work if you're the only person in your company who will be using it.

That's because, fundamentally, agile is a team-based approach. If you want to apply the agile methodology to any undertaking in your organization—agile development, agile roadmapping, or agile project management—that effort is going to require the participation of a cross-functional team operating under a unified philosophy.

If your development team insists on a waterfall approach, they're going to demand detailed documentation for everything you want for your product before they start work. Not much you can do about that.

If your executive stakeholders have the waterfall mindset, then before they give you the green light and development resources you need, they're going to want to know exactly what you have in mind for the finished product—not the minimum viable product you'd like to release so you can gauge what resonates with early adopters. Not much you can do about that, either.

So the bad news is, if you're an agile-minded product manager, and you can't get any of the key players in your company to adopt that philosophy, you almost certainly won't get very far implementing your agile approach to product development.

The good news is you don't need to make agile product management work alone. You already have the skills required to encourage the broader adoption of agile throughout your organization. As a product manager, you are by definition an expert at leadership, persuasion, and strategically moving a cross-functional team toward a shared goal.

So here's the question: If you're a believer in the agile product management approach, and you find yourself responsible for a product in an organization still dominated by a strictly waterfall mindset, what can you do to evangelize the agile mentality and drive adoption of agile principles?

EVANGELIZING AGILE IN AN WATERFALL ORGANIZATION

It is not easy to make sweeping across the board changes to not just your processes but also the underlying mentality behind why things are done the way they're done. This is especially difficult if your organization doesn't know much about the agile philosophy or the benefits of embracing it. Your first step is to become an agile champion and educate your organization on what agile is and why it's so wildly popular today.

But, if despite your best efforts to be a champion for agile product management, you still get some resistance from your team, consider taking another more subtle route.

BE CREATIVE: DON'T CALL IT AGILE

Asking your entire organization to make a wholesale change to the way they approach product development would be a tall order. You'd be proposing not only that your cross-functional team make sweeping changes to how they operate, but also that they adjust their entire approach to designing and building products.

So don't call it agile. You can talk up specific benefits of the agile methodology and gradually introduce agile product management principles to your organization without ever referring to them as agile. No need to alert everyone that you have a larger agenda to bring them around to an agile philosophy.

BE TACTICAL: SWITCH TO THE RIGHT ROADMAP TOOL

Remember, a key difference in how waterfall and agile organizations operate is

in how they use their product roadmaps. A waterfall-oriented team develops its roadmap as a long-term, detailed, set-in-stone plan—and then follows that plan linearly, no matter what.

An agile roadmap, by contrast, reflects the agile company's understanding that strategies will change, priorities will change, and real-world user feedback can and should affect how a product is built.

AN AGILE ROADMAP REFLECTS THE AGILE COMPANY'S UNDERSTANDING THAT PRIORITIES WILL CHANGE.

This is one reason waterfall product teams can get away with using static tools to build their roadmaps. These applications (presentation software, spreadsheets) aren't ideal for roadmaps in any environment. The need for manual updates, the difficulty involved in sharing documents, and the version-control issues that inevitably arise, all make them suboptimal choices for housing your roadmap. But for a waterfall team that doesn't plan to revisit its roadmap more than once a year, they can provide a workable solution.

For an agile team, on the other hand, having the right roadmap tool—ideally a web-based, easy-to-build and easy-to-share application designed specifically for roadmaps—is essential. Building your product roadmap with the right tool will also make it much easier to subtly bring your team around to the agile mindset.

When you hold product meetings, with your web-based roadmap projected on the wall (or shared via your screen), you can make updates to the roadmap in real-time. Just showing your colleagues how easy it is to make little adjustments to the product's strategic plan will have the subtle effect of convincing them that they don't need to rigidly adhere to a plan written down eight months ago.

When everyone can easily view your product roadmap online, and see when and how small tweaks have been made to the strategy, that will also have the positive psychological effect of making the original, documented plan feel less like it has to remain set in stone.

So if you want to persuade your organization to transition to an agile company, choose your product roadmap software wisely.

BE STRATEGIC: START SMALL

Finally, remember that agile is all about chunking down, not up. So why not apply this approach to your plan to introduce your organization to agile product management?

Start with small adjustments. Shift from documenting product features to describing user stories. Make yourself more available to your development team. Share user feedback and other relevant data with your executive stakeholders more frequently.

These are all agile principles, of course—but you're not likely to receive much pushback on any of them. In fact, your teams are likely to embrace them all. And before you know it, you'll be an agile shop!

APPLYING WHAT YOU'VE LEARNED

There's no magic formula for success as a product person in an agile environment. At the same time, the lack of a formula or rigid framework is part of what makes agile work for so many teams. Agile values and principles serve as guides rather than prescriptive rules and restrictions.

All you need to get started down a more agile path is a desire to learn, a willingness to embrace change, and a drive to continuously improve. Sounds suspiciously like the introduction to a self-help book, doesn't it? That's because that's kind of what we've written here: agile is what you make it, and yes, it can improve your life.

The real benefits of agile do not come from adopting Scrum or any other agile framework, but from truly understanding the principles and applying them to build better products.

A FEW WORDS FROM THE EDITOR:

Ship It! is the product of research, observations, and conversations with product managers across a variety of industries, and the shared experiences and learnings from various members of the ProductPlan team.

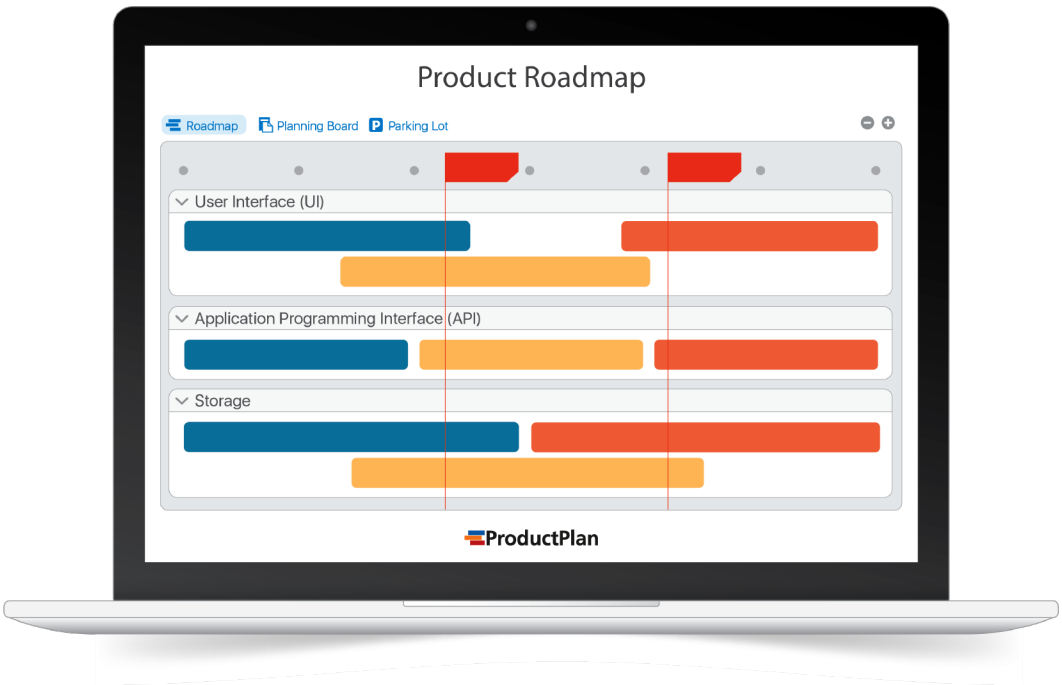
Our team wrote this book because we felt a guide to agile, written specifically for product people, was long overdue. Our goal is to empower product people from organizations large and small to leverage the values and principles of agile product development to achieve great things.

Whether you're a product manager, product owner, or something in between, I sincerely hope our first edition of Ship It! has given you the knowledge and tools needed to be successful in agile product management. But, in the spirit of practicing what we preach, I'd like you to know that by no means do we equate "shipped" to "done," in regards to the content we publish.

We learn new things every day. And so do you. So let's continue the conversation. I'd love to hear your thoughts on the concepts we've discussed here, but more importantly I'd like to hear about your own experiences as an agile product person to help inform our next iteration.

Heather J. McCloskey
heather@productplan.com

TAKE THE FIRST STEPS TOWARD AGILE PRODUCT MANAGEMENT



TRY IT FREE →

A DYNAMIC PRODUCT ROADMAP TOOL MADE FOR AGILE TEAMS.

ProductPlan is the easiest way to plan, visualize, and communicate your product strategy. We believe roadmaps are essential to create organizational alignment and ship successful products. Our intuitive features for building, managing, and sharing roadmaps help teams across the globe convey the big picture in one place.