MASTERING PRODUCT DEVELOPMENT

productfolio

TABLE OF CONTENT5

1. Introduction 3

2. What is Product Development? 4

3. Roles in Agile/Scrum 10

4. Creating a Feature 17

5. Writing Requirements 23

6. Managing the Backlog 32

2. Conclusion 39

We’re so glad you’ve downloaded this ebook! Let us quickly introduce ourselves before jumping in.

Productfolio creates amazing Product Management software, so your team can focus on creating amazing products. We’re passionate about the craft of Product Management and have learned from interviewing hundreds of Product professionals, that every team is a little bit different, but most successful teams share a core set of patterns and principles.

We hope these insights are helpful and love to hear from you if you have feedback or questions about this content herein, or our platform (hello@productfolio.com).

Happy Product’ing!

Neal

productfolio

1. WHAT IS PRODUCT DEVELOPMENT?

WHAT IS PRODUCT DEVELOPMENT?

Product Development is a complex topic that has evolved over the past couple of decades. We'll review those processes, the most common standards used today, and the roles and responsibilities of Product and the rest of the team.

Once we've covered these concepts, we'll walk through the discovery, development, and deployment steps of developing a new feature, to give a sense of how it all comes together. Let's get started!

DEVELOPMENT PROCESS

Software development processes has evolved over the past 50 years, in part learning what works best for developing software, and because the delivery model has shifted from products on shelves, to online delivery, that enable a more incremental approach.

In the 1960s, Waterfall methodology was the dominant approach. There was a series of distinct "stage gated" steps and one must be completed before proceeding to the next.

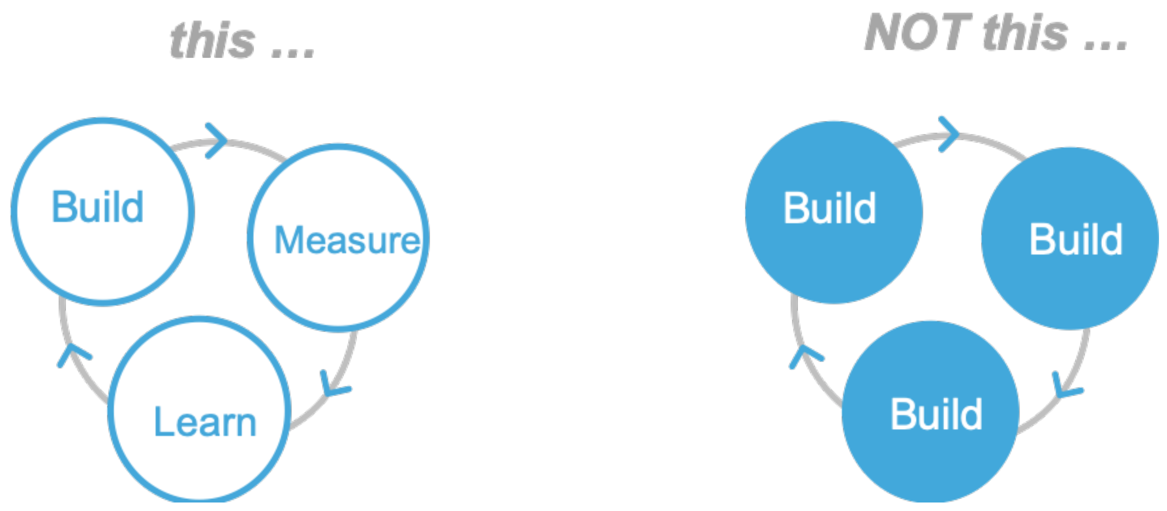
First requirements needed to be completed, then design, then development, validation, and finally release. The approach resembles an assembly line that can maximize the efficiency of resources when developing a known commodity.

Unfortunately, there's also a downside – because of the stage-gated approach, practitioners and project managers would try to minimize having to go backwards in those steps, leading to perfectionism and trying to account for every detail before proceeding to the next stage. This inevitably slowed the whole process down and increased the cost for getting anything completed.

In the 1990s, a number of contrarian processes like Rapid Application Development (RAD), and Extreme Programming (XP) began to emerge, which proposed another approach to development that was more lightweight and iterative in nature. And in 2001, a number of software engineering leaders met on a retreat to establish the Manifesto for Agile Software Development.

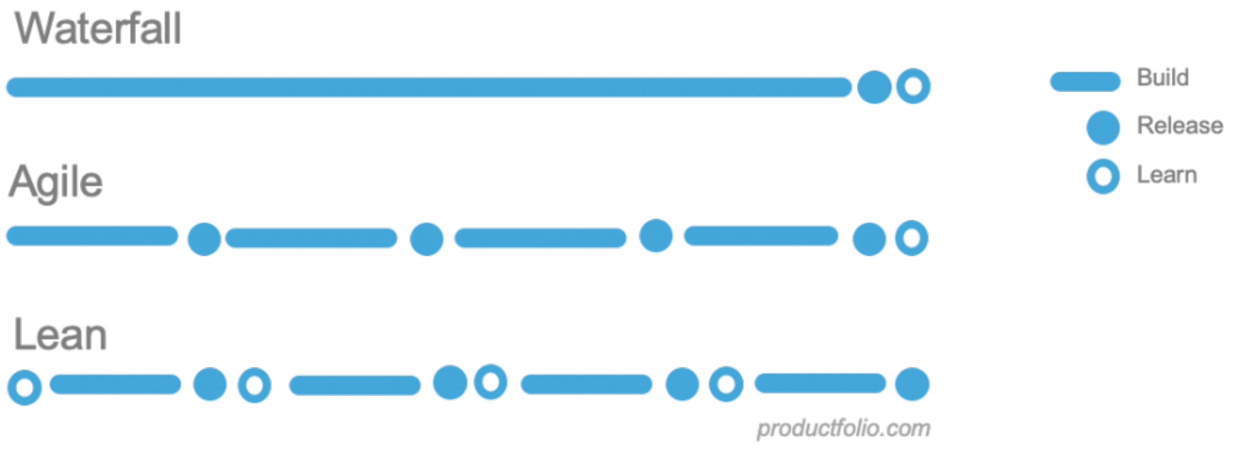
This was the beginning of Agile Development, which put people over process, working software over documentation, and responding to change over simply following a plan.   In the subsequent years, Scrum became the most common Agile Project Management process, and teams begin working in 2-week increments, and learning how to develop requirements and design, in parallel to development and release efforts.

A decade later in the 2010s, Lean product development rose to popularity and built on top of Agile for those just learning about it. Whereas Agile is about *developing* increments of software and delivering them quickly, Lean is about *discovery* increments and provides a methodology for efficiently achieving product-market fit.



Lean introduced the idea of a Minimal Viable Product (MVP) and espoused the goal of getting to product-market fit as quickly and efficiently as possible, with minimal waste. This is a natural complement from the Product discovery perspective, to Agile in the product development perspective. The rapid build>measure>learn loop provides a recipe for discovering and building value through an Agile development pipeline.

Today, there are any number of combinations of these processes in practice across the industry. For the most part the industry has shifted toward Agile thinking, though many organizations will practice some hybrid of Agile and Waterfall (aka “Wagile”). There are even frameworks for formalizing that hybrid like the Scaled Agile Framework (SAFe).

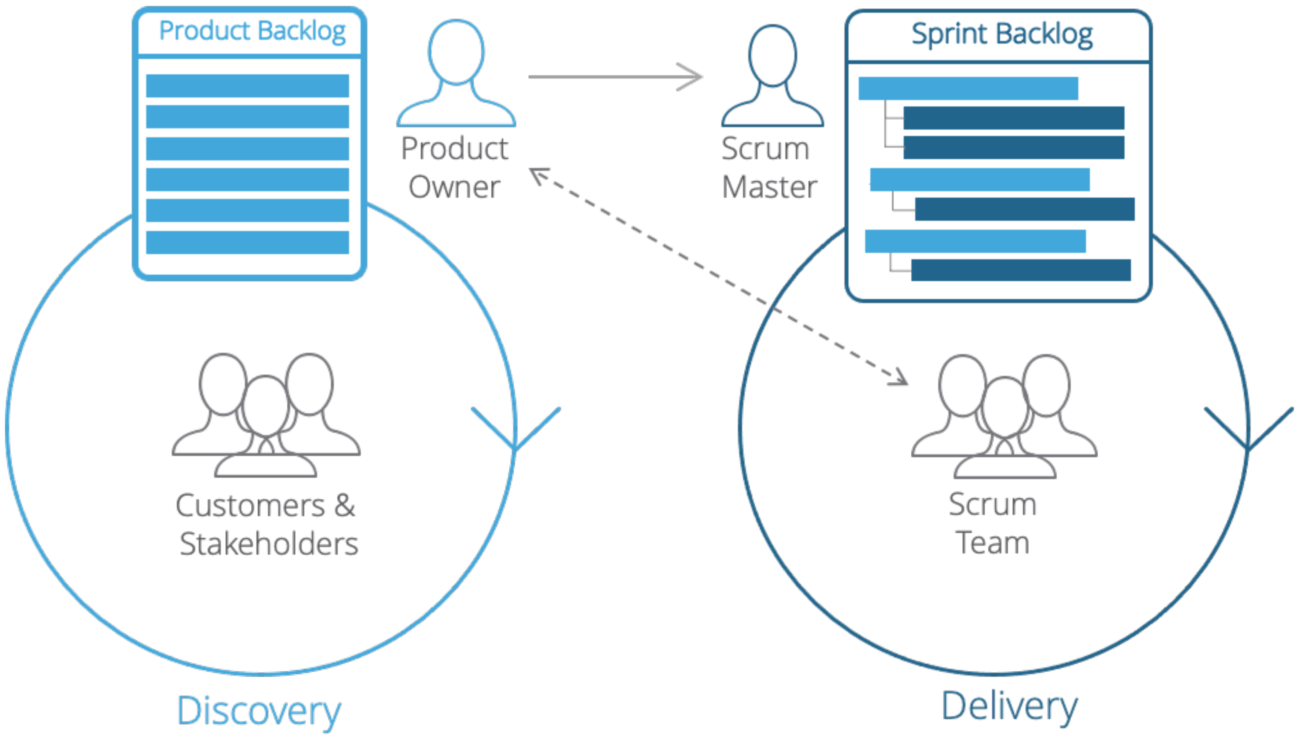


The truth is, there is no perfect process, and each one of them has their strengths and weaknesses – it’s just a question of what we want to optimize for. Since most organizations do recognize some flavor of Agile/Scrum as the basis for their product development, we’ll use that as a point of reference for the rest of this discussion.

2.
ROLES IN AGILE / SCRUM

ROLES IN AGILE/SCRUM

There are three key roles to know about in Agile/Scrum: the Product Owner, ScrumMaster, and the team. The Product Owner "owns" the backlog, providing requirements and priority to the team.



THE

PRODUCT OWNER

The Product Owner works with one foot outside of the team, understanding needs, opportunities, and priorities, and then communicating that back to the team.

In the context of Product Development, the Product Owner is often a member of the Product Management team and thus spending a good amount of time working with customers and business stakeholders to understand all aspects of the functional outcome that must be achieved with the solution the team is working on.

This is often expressed as a collection of requirements, stated in the form of User Stories, and accompanied by a set of Acceptance Criteria that must be met by the solution, which is designed by the team.

Put another way, the Product Owner will represent the problem, and the cross-functional team, will design the solution.   In this context, it is sometimes said that Product works in the "problem space" whereas the team (Engineering, UX, Data, etc) work together in the solution space.

The Product Owner is responsible for determining the “what & why”. The team, led by the ScrumMaster, owners Delivery and determines how to solve the problem and how long that should take. Release/launch plans are determined together by the Product Owner and ScrumMaster, based on the desires of Product and the capabilities of the team.

SCRUMMASTER

The ScrumMaster is the administrator, leader, and protector of the team. They ensure the team follows best practices, updating tickets, ensures the team is working efficiently toward priorities, and not committing too much.

The ScrumMaster can be anyone on the team, though it is often an Engineering team lead. Or, when the team is large and truly cross-functional with a lot of non-engineering capabilities represented, there may be a Program Manager in that case who plays this role and facilitates the team, and ensures Scrum best practices.

A Program Manager is a process and delivery efficiency expert, who may play this role for multiple teams within a focus area, keeping the teams coordinated, aligned, and running smoothly.

TEAM

The team is composed of a group (ideally 5-8) of cross-functional members that span all of the capabilities necessary to complete the requirements from Product. For example, if the team is building a SaaS product, you'd ideally have WebApp developers, mobile app developers, UX, and QA on one team.

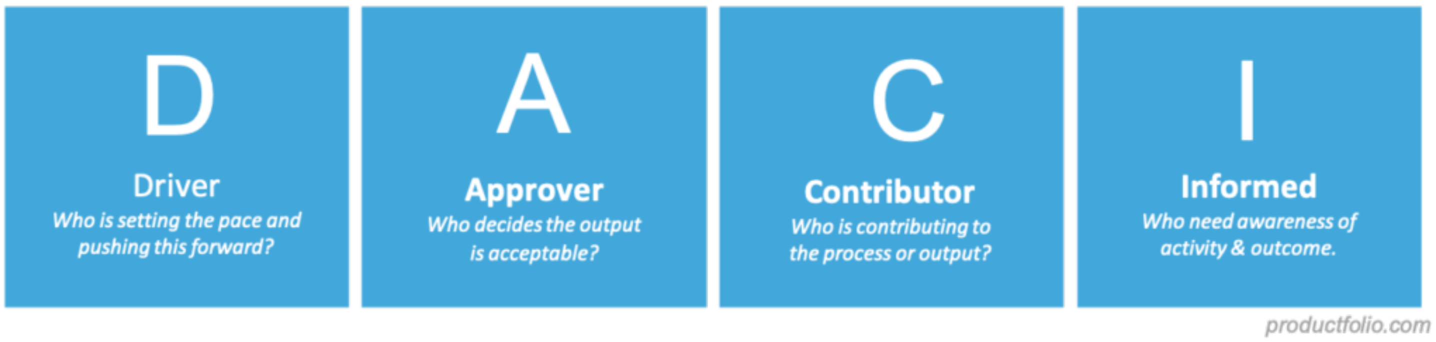
In the case of a larger team working on a more sophisticated application, you might have multiple scrum teams each working on specific features of the application – these teams are sometimes referred to as feature teams, and similarly you would have the capabilities represented that are required for that feature.

If the team was working on the search & personalization algorithms then you might have a data scientist on the team and API developers rather than WebApp and UX.

RESPONSIBILITY OF ROLES

The strength of Agile development is flexibility and agility but this can also sometimes be its weakness. Everyone interprets roles and responsibilities a little bit differently which can lead to confusion, gaps where things aren't accounted for, and overlaps that can lead to toe stepping and politics.

It can be really helpful to discuss expectations about roles and responsibilities as a team from the outset, to avoid such issues and make sure expectations are aligned.

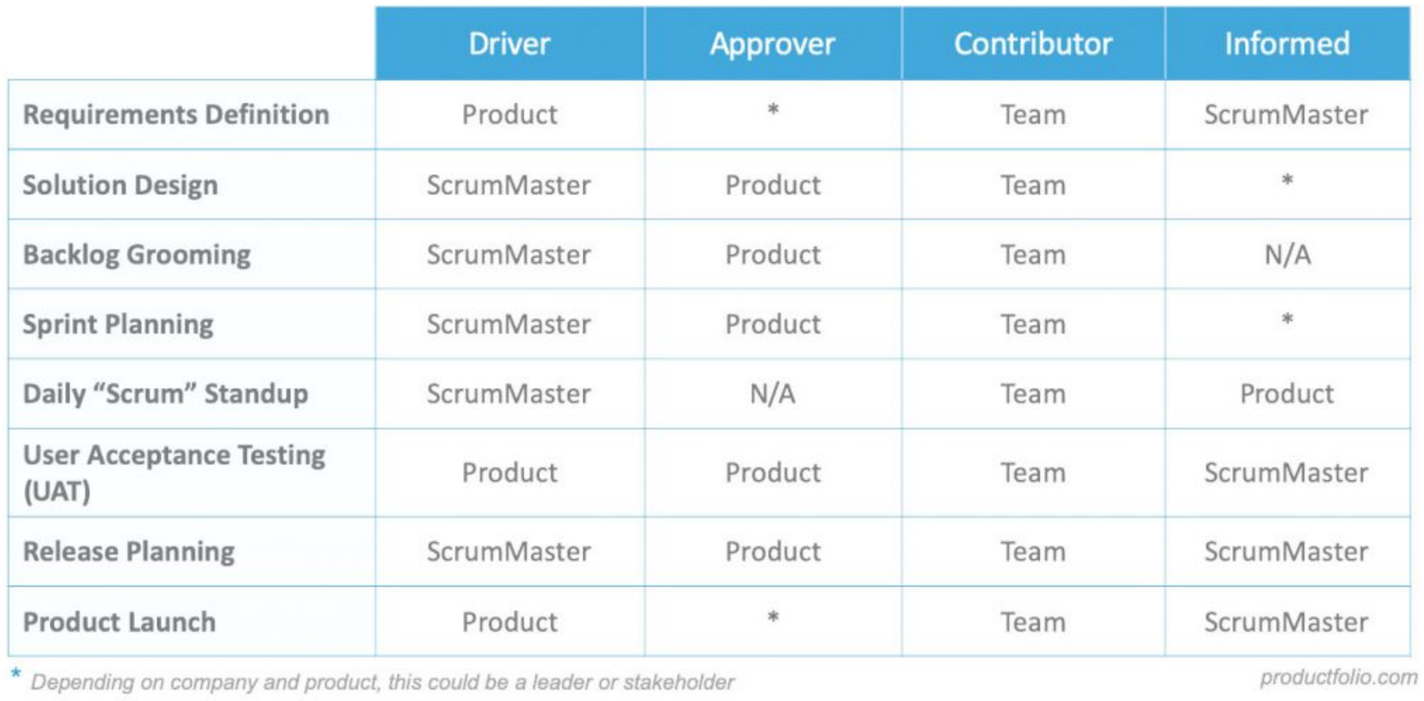

D
Driver
Who is setting the pace and pushing this forward?
A
Approver
Who decides the output is acceptable?
C
Contributor
Who is contributing to the process or output?
I
Informed
Who need awareness of activity & outcome.
productfolio.com

The DACI model is a useful framework for aligning those expectations. DACI is an acronym and stands for Driver, Approver, Contributor, Informed. The goal with this framework is to identify activities of the team, and map the responsibility for each of these activities, for each of the roles.

For example, the Product Owner is Driving requirements definition, though the team may contribute insights to Product where it is helpful.   There shouldn't be anyone on the team who needs to Approve the requirements, though it's possible a Business or Product leader outside of the Scrum team may practice some level of oversight.

And, the ScrumMaster would be kept informed of the requirements as they enter the Scrum team, so they can manage the team's activities against them, accordingly.

Here is an example table that maps the common activities and responsibilities as reference.



	Driver	Approver	Contributor	Informed
Requirements Definition	Product	*	Team	ScrumMaster
Solution Design	ScrumMaster	Product	Team	*
Backlog Grooming	ScrumMaster	Product	Team	N/A
Sprint Planning	ScrumMaster	Product	Team	*
Daily "Scrum" Standup	ScrumMaster	N/A	Team	Product
User Acceptance Testing (UAT)	Product	Product	Team	ScrumMaster
Release Planning	ScrumMaster	Product	Team	ScrumMaster
Product Launch	Product	*	Team	ScrumMaster

* *Depending on company and product, this could be a leader or stakeholder*

productfolio.com

3. CREATING A FEATURE

CREATING A FEATURE

Now that we have established processes, roles, and responsibilities, let’s walk through the process of creating a new feature to see how it all comes together.

Assuming we have already defined the feature requirements (User Stories), then we’re ready to engage the team on the 3 high-level stages of feature development – Discover, Develop, and Diagnose.

DISCOVERY

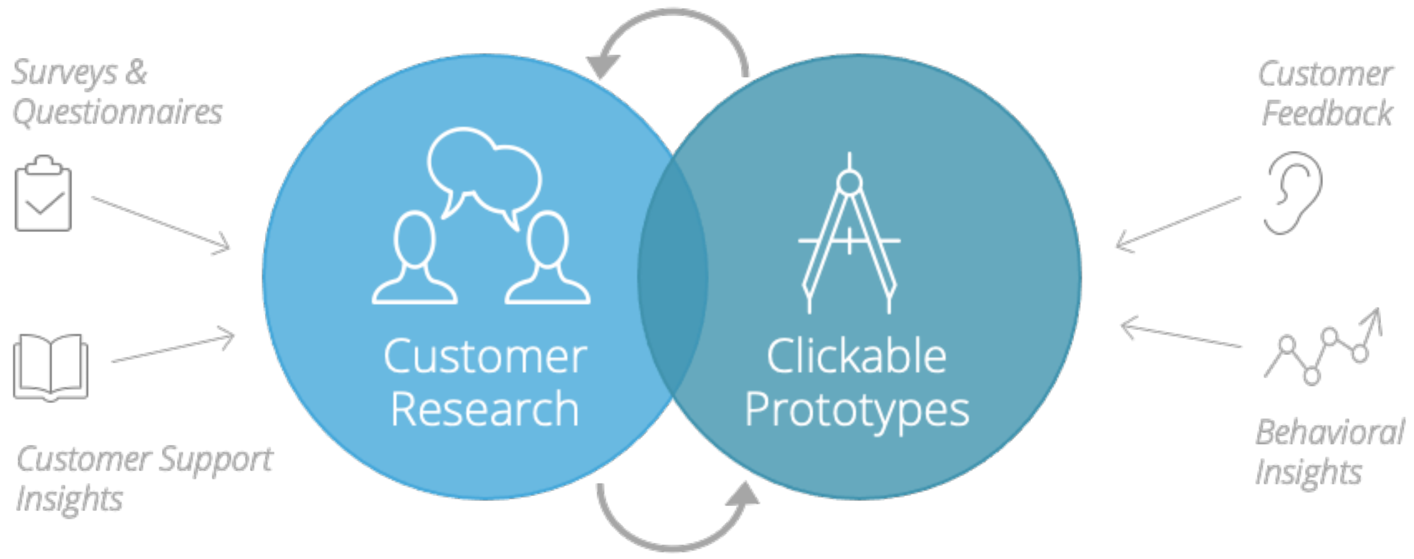
The product requirements that we wrote for this feature, in the form of User Stories, describe the problem we need to solve, and include acceptance criteria that indicate the bounds of an acceptable solution.

We intentionally take this approach and avoid prescribing a solution in the requirements, in order to facilitate solution design by the team – that process begins with learning more about the context of the problem, in order to design an effective solution.

In the case of a workflow feature, Product may partner with the UX designer to conduct user research, observe the difficulty, and validate prototypes of a possible solution.

Or, in the case of a search algorithm, Product may partner with the Data Scientist on the team to review data and better understand the insufficiency of the algorithm.

In both cases the approach is similar – Product partners with the capability expert on the team to do discovery and better understand the problem, handing off to the team member(s) to design the solution, and approving the design once completed.



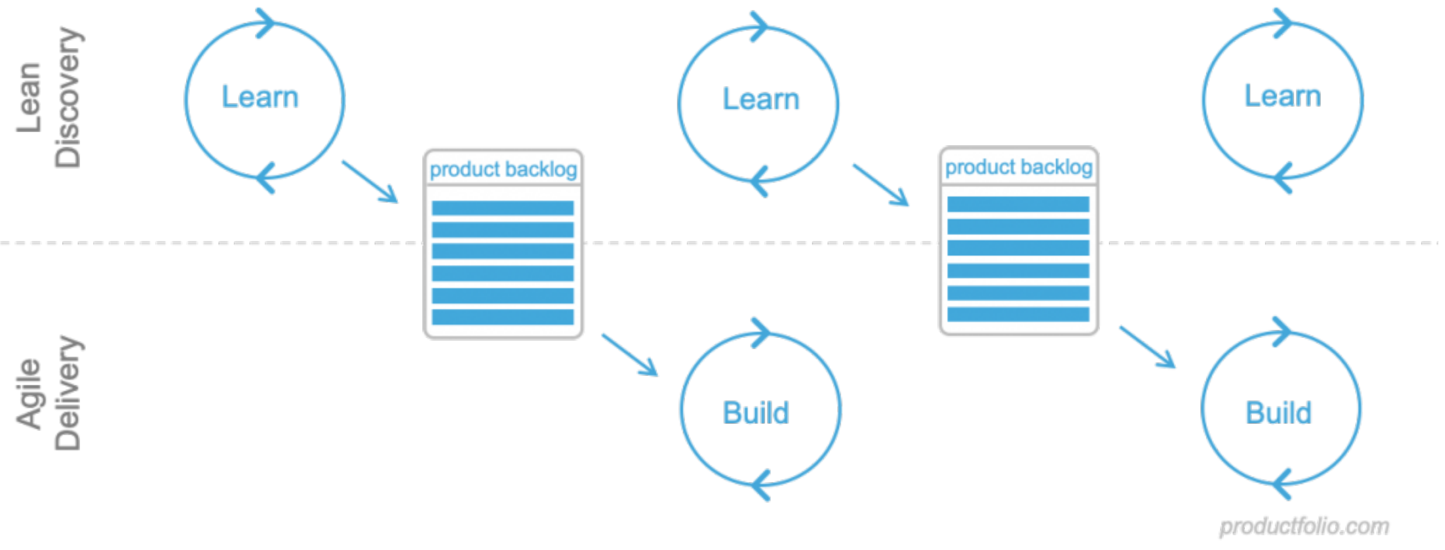
DEVELOPMENT

Once Discovery is complete, the User Story is taken to the next backlog grooming session where the team will break down the solution into sub tasks that must be done in order to build that solution.

The Product Owner supports this process by answering questions and providing clarity where it is needed, and working with the team to determine the release plan for the feature (timeframe, A/B testing, etc). From there, the Product Owner becomes more passive until we’re nearing time for feature deployment.

The primary responsibility of the Product Owner during the Development stage, is Discovery on the next feature, so that it is ready for Development when the team is done developing this feature.

In Agile/Scrum process, Discovery can be thought of as a track of work that happens in parallel to Development – a concept sometimes referred to as Dual Track Scrum.



PROCESS ANTI-PATTERN

A word of caution – it can be tempting to insert yourself into the day to day activities of your team during development; more than one Product Owner has succumbed to day-to-day need for administration and coordination by the team.

Don't do it! These are the responsibilities of the ScrumMaster and spending your limited time here may seem like the right thing to do, but it will abdicate your core Product responsibility and earn you more responsibility as a Project/Program Manager, rather than Product Manager.

Remember, your core responsibility is to ensure your features are useful, usable, and valuable to the customer.

DEPLOY

Whenever possible, features should be deployed as a limited availability test (eg Alpha, Beta), so impact can be measured and evaluated before full availability of the feature.

When the feature is not performing optimally, or in the case of an MVP release, this can lead to a sub-stream of additional work in the coming sprints that we'll balance with other feature development, in order to ensure we perfect the feature before calling it 'done'.

Remember that as Product, we measure success by the outcome of our work against our objectives & KPIs, not merely the delivery of a feature.

4. WRITING PRODUCT REQUIREMENTS

WRITING PRODUCT REQUIREMENTS

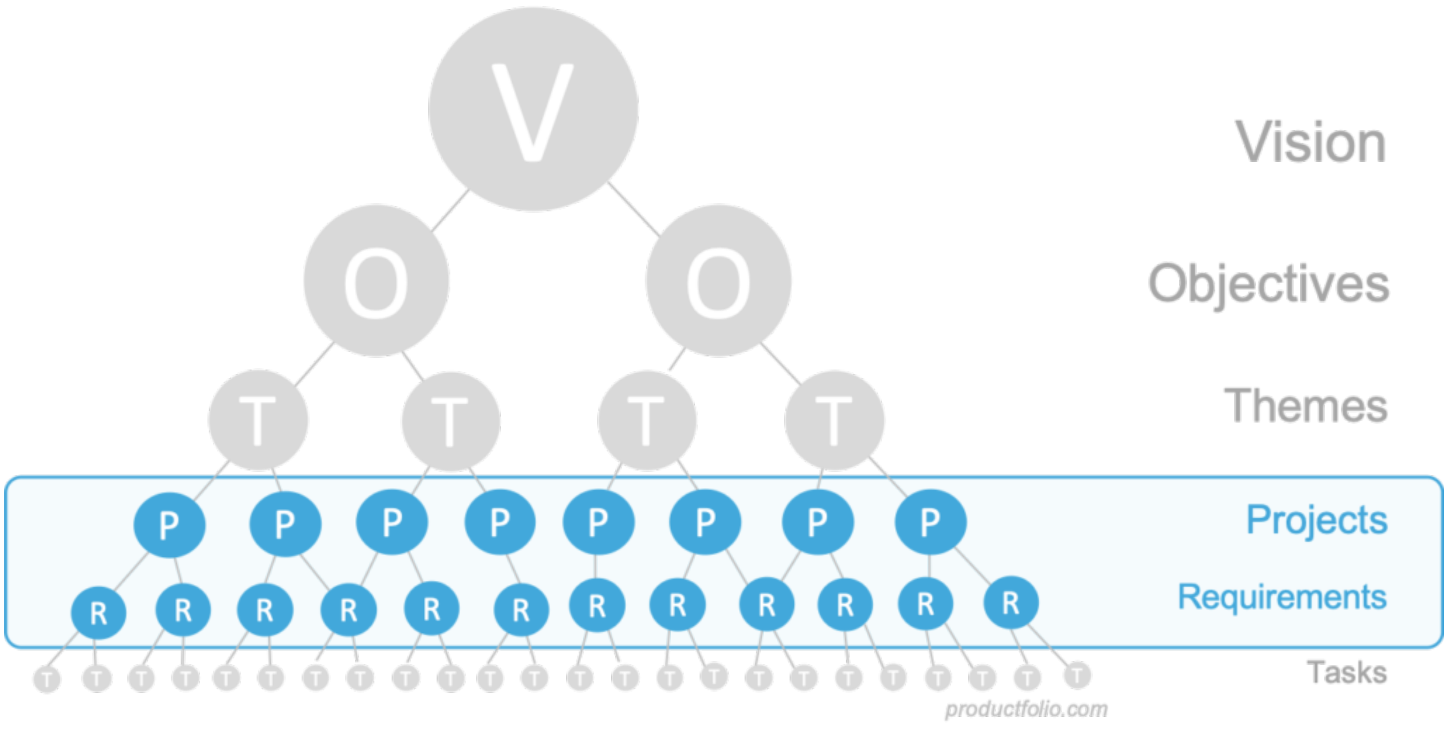
Now that we’ve covered process, roles and responsibilities, let’s dive into a key deliverable of the Product role:

Requirements provide an outline of an acceptable solution for the team that is building features for the product. Designers create an experience, Engineers develop systems, and Data Scientists create predictive models to solve the problems that have been prioritized and defined by Product.

When done well, a requirement is not prescriptive – Product works in the “problem space” to understand the problem and what a viable solution should be, and it Is the team who works in the “solution space” to then design that solution to the defined problem.

Requirements define specific aspects of a solution, Product or Feature. More often than not, a Requirement describes a dimension of a new feature the team is working on for the Product.

Requirements can either be ‘loose leaf’ and describe small improvements to existing functionality, or can be part of a set of requirements defined in a project (aka product brief).



WHAT IS A PROJECT?

A project is something typically 1-3 months in scope that might be prioritized on a roadmap. In Agile/Scrum terminology this might be called an Epic.

During roadmap planning, we might call this a roadmap candidate since it is being considered for prioritization and thus inclusion on the roadmap. Once it is approved as part of the plan and becomes an actionable piece of work however, we begin referring to it as a Project (or Epic).

There are two types of projects generally – Feature Projects and Capability projects. Feature Projects are useful and usable to the user, and Capability Projects introduce new capabilities to the platform/system, that future features can be built upon.

For the most part in Product Development, you'll be working on Feature Projects, but it is good practice to work with your technical partners and set aside bandwidth for those enabling capabilities.

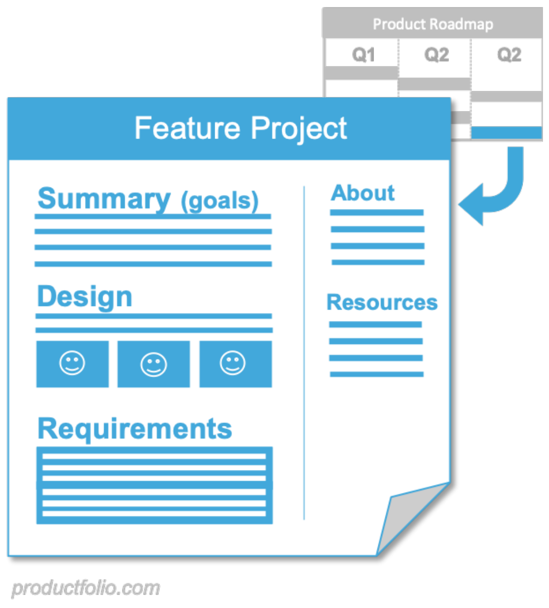
PROJECT DEFINITION (A PRODUCT BRIEF)

Unlike how most Scrum teams use Epics as merely containers for related User Stories, Product Management needs to do real work to define the feature at this level.

A feature is more than just a collection of requirements – there is context and purpose that is greater than the sum of its parts, and articulating that is helpful both for clarifying one's thoughts and communicating that to the team for alignment.

As Dave McCullough eloquently put it "Writing is thinking. To write well is to think clearly. That's why it is so hard".

And so, let's think about this in terms of defining a feature. We'll want one document to serve as a "single source of truth" reference doc, that the team can easily find and reference. We'll want to keep it concise but offer the following basic information:



PRODUCT BRIEF, NOT A PRD

If we go back to Waterfall development in the 1970s, it was common to have a very thorough Product Requirements Document (PRD) created before any development work could begin – and that's still the case for many physical projects in fact such as buildings or rocket development.

It makes perhaps sense when we have high confidence in what needs to be built and the cost of mistakes are high.

But this is not how modern software development works. Because it is so easy to push updates out to users, it is very easy to deploy variations to test and learn, as well as it is is easy to fix issues. For this reason, most software development has moved away from Waterfall and toward Agile.

That doesn’t mean there is no value in crystallizing the context and purpose at a feature project level however – quite the contrary!

The answer is to find the middle-ground with a lightweight “product brief” that provides context for the team and concisely captures the key points as a way of defining the problem space (goals, needs, high-level requirements), but without holding things up since a lot of discovery can be done concurrent to development, and not encroaching upon the team’s agency to create a solution.

Here’s a simple list of content you might consider including in your product brief:

- **Summary** – a paragraph synopsis synopsis.
- **Discussion** – a deeper conversation about the nuances, logical rules, and context of this feature.
- **Goals** - human language definition of the outcomes and metrics for measuring success.
- **Design** – UI flows and mocks that describe the user experience for the solution. Note – this may not be in the initial iteration of the document but something you layer in after working with UX.
- **Requirements** – list of high-level requirements / user stories that define the bounds of a solution.

TYPES OF REQUIREMENTS

In traditional Business Analysis, there are many types of requirements – Functional Requirements describe a technical perspective, Non-Functional Requirements describe an internal process such as QA against those functional requirements, Business Requirements describe an expectation of internal business stakeholders, and yes, User Requirements which describe external user needs.

USER STORIES

Consistent with the philosophy of Agile, User Stories are intentionally non-prescriptive for the team, and merely describe a scenario to inform the team during solution design.

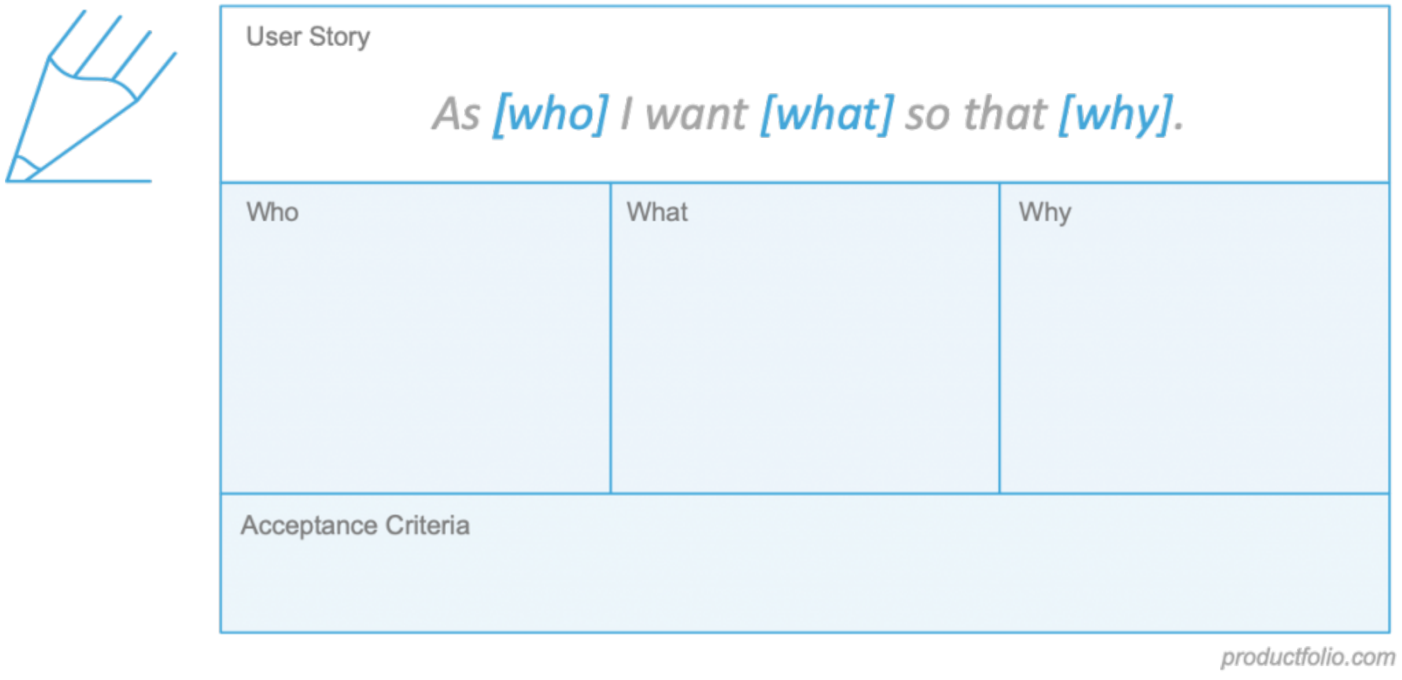
They are usually written in the form of: *“As a [who] I want [what] so that [why]”.* For example, *“As a customer I want to see recent orders in my account so that I can track the order and know when it will arrive.”*

Optimal User Stories follow the INVEST principles which means they’re independent from one another (no dependencies), independently negotiable and valuable, specific enough that the team can estimate them with story points, small enough to complete within a single sprint and testable/verifiable through QA processes, that are typically keyed from your acceptance criteria.

In addition to a user story, some choose to include a acceptance criteria for additional depth.   Acceptance criteria often a simple bulleted pointed list of details that must be true for the solution to be “accepted” and the User Story to be marked as done.

Using that same User story example, we might write acceptance criteria that say: (i) must include purchase mount, (ii) must include all orders from past 24 months, and (iii) must indicated if order was fulfilled or is still out for delivery.

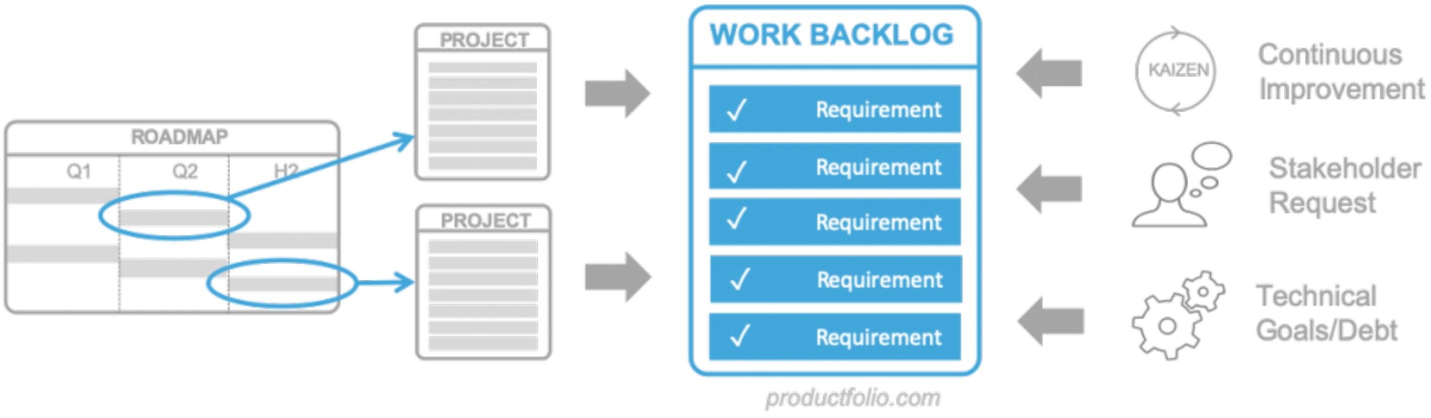
The following template can be useful for defining these requirements.



MANAGING THE BACKLOG

The Product Backlog is where all the requirements come together for prioritization. Once you've created a couple of projects, you might have a dozen or two requirements from those sources.

That's not the only source of requirements though – they can come in the form of "loose leaf" continuous improvements, A/B testing hypotheses, support requests from stakeholders, technical stories to refactor, or any number of other sources. Ultimately all of these requirements come together in a single backlog for prioritization.



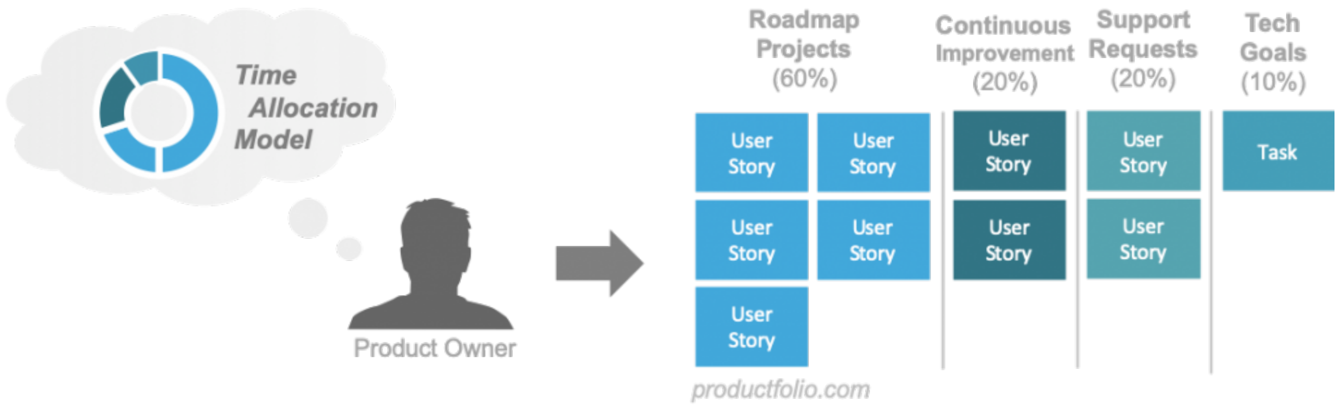
6.

MANAGING THE BACKLOG

PRIORITIZING THE BACKLOG

It is good to have a time allocation model in mind, that budgets your time for the upcoming sprint. For example, a good rule of thumb might be to spend 60% of your time on those projects that are on your roadmap, and set aside 20% for continuous improvement activities such as A/B tests and refining the UI, 10% for tech debt or other tech initiatives, and 10% for support activities such as stakeholder requests.

The time allocation model is going to be different for every organization, but defining and aligning on that ahead of time will prevent a lot of difficult conversations down the road as well as ensure you're staying true to your longer-term goals and responsibilities.



RELEASE PLANNING

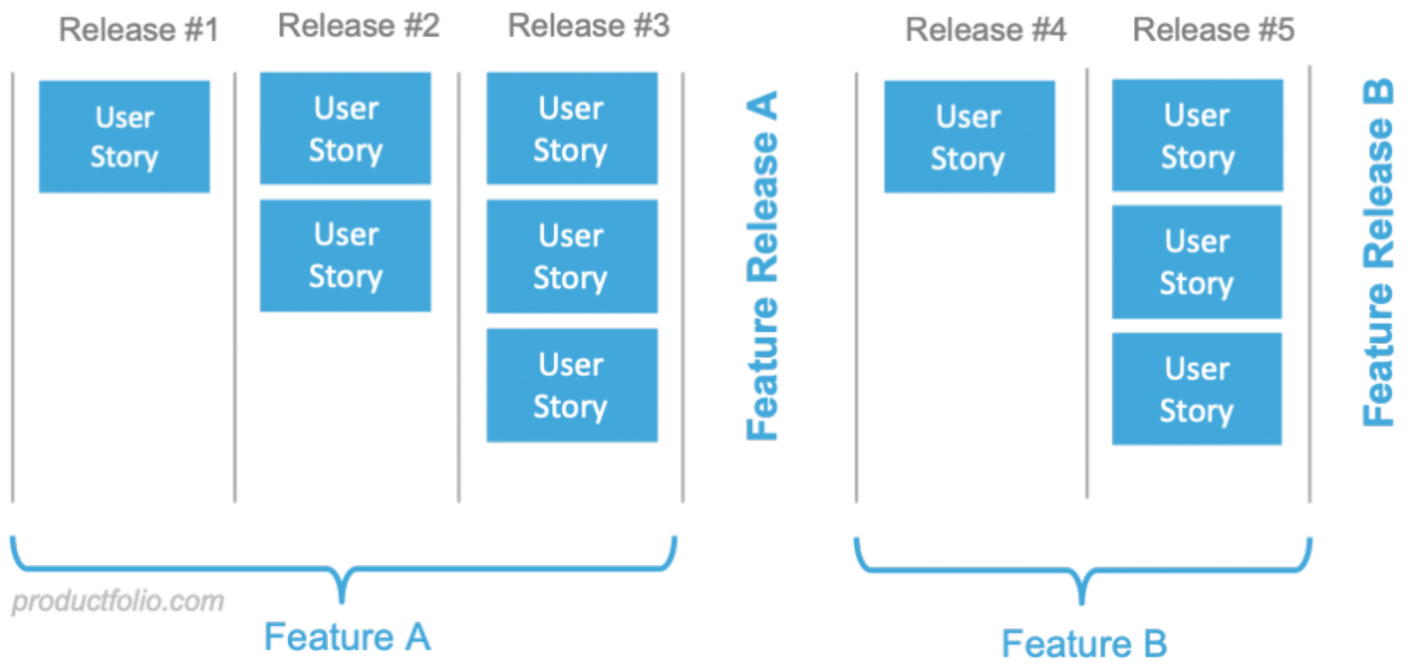
Release planning is the process of determining how a new feature will be made available to the user. Since features are meaningful functionality for the user that is useful, usable, and valuable to the customer, it is often an aggregation of requirements user stories that spans multiple sprints. As such, Product needs to work with its technical partners to devise a plan for how to release this feature.

Every team has a little bit different release process – some use a “release train” every few weeks in which all features that are for a release must be readied on the same cadence in order to catch that next ‘train’.   In other cases, a team may practice Continuous Deployment, meaning Stories are released incrementally, as they are completed.

In both cases though, completion of an individual requirement / user story doesn’t necessarily mean the full feature that is valuable to a customer is ready to be released for use. Internal teams can also be caught off guard and not ready to support the new feature. Not considering this and indiscriminately releasing every increment directly to the customer can be frustrating for everyone involved.

A better approach is to implement a “feature flag” that suppresses the feature until everything is aligned to provide the full new increment to the customer. Technical teams can deploy code in whatever manner works best for them and we simply “flag on” the feature when it is ready.

To this end, it is wise to plan feature release in advance, so the user stories can be lined up properly in your development sprints that follow. This way, every sprint may user stories for different sub-feature functionality – and the team can deploy the code at the end of every sprint, if that’s the desire … but we can hold back the release of a feature until all the stories for a given feature are completed.



REQUIREMENTS ARCHIVAL

Lastly, it is important to keep track of releases that have completed and the requirements in each of those, in case you need to figure out what released in the future.

People can forget why the system behaves the way it does, and regression testing can omit these requirements from coverage.

For this reason, it is important that legacy requirements be discoverable and reference-able after they have gone live. This enables Product teams can do research and QA teams can link their tests cases to these requirements.

Teams can also work together on solutions to de-couple the code release cycle from release of features to users, to ensure those larger features that require multiple sprints, are ready before they're enabled. Product Management plays an active role in all of this, when playing the Product Owner role for their Scrum team.

CONCLUSION

Agile/Scrum has become the most popular process for product feature development. The role of the Product Owner in Agile/Scrum is often played by a member of the Product Management team who works closely with the product development team(s) to discover, deliver, and optimize features.

The Product Owner plays an active role by writing requirements that frame the problem/need, often in the form of user stories, so the team can create an acceptable solution.

PRODUCT MANAGEMENT SOFTWARE

Whether you're a new Product Manager learning the role or a team leader, seeking to standardize planning for your team, Product Management software can streamline the process of planning and enable your team to focus on building great products. Check out how Productfolio can enable your team and elevate the craft of Product in your organization.

productfolio

We create amazing Product Management software so your team can focus on creating amazing products.

Try 100% free for 14 days. No credit card required.

TRY FOR FREE