

The Leading Reference for technology-based products

# Product Management Journal

Volume 7 £10 / €12 / \$14

Updated

## Contents

### Agile

Everything you need to know about Agile p04

### Scrum

What it is and how it works p11

### Focus

How to survive and prosper with Agile p20

### Discovery

Tools to build market insight and customer empathy p23

### Agile has to scale

We explain the different approaches p30

### Tools

Software to help you manage requirements p39

### A guide to Zen Agile

A guest blog with bite p40

### Product Management / Agile fit

How does product management fit with Agile? p49

Insight: **What's the point of Product Management in Agile?**

You still need product leadership p58

# Agile

And its impact on product people

product

focus

# Training

## Product Management and Product Marketing

Industry-leading public and private courses  
Live online and face-to-face delivery



Focus on technology-based products  
Tools, templates and checklists  
Learn best practice  
Certification



product

focus



[www.productfocus.com](http://www.productfocus.com)

# Welcome

Leading the way for Product Managers

## So what does Agile and Scrum mean for product people?

You may be working in a software company that has always used Agile development approaches like Scrum. You might be encountering Agile for the first time and wondering what value it can bring to your business. Maybe you've been asked to take on the Product Owner role.

Our aim in this Journal is to give you a basic understanding of what it's all about and importantly, what it means to you as someone managing products.

Agile is now established as the dominant approach when developing software products. While there has been widespread adoption and undoubted benefits, the traditional Waterfall approach is still used in a vast number of organizations. We want to sift the hype from the reality and help you talk sensibly about Agile.



## Who's who?

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

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

To get all our previous journals, and receive the latest copy, sign-up at [www.productfocus.com](http://www.productfocus.com)



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

# Agile

Everything you need to know about Agile

“When you have an Agile mindset, you accept the fact that you face uncertainty. You approach things in a way that allows you to continuously learn and adapt. You seek to remove that uncertainty.”  
**Kent McDonald,**  
**Product Manager**  
**& Founder,**  
**KBPMedia**

**Agile was born in the world of software development, where you can build the first version** and keep adding to it or changing it to make improvements. You get to see results more quickly, and because you can check and change what you do in each iteration, it lowers the risk of getting it wrong. It is an approach that accepts uncertainty and promotes experimentation and close collaboration with customers.

The Agile approach is often contrasted with Waterfall methodologies, where development flows through a series of phases that can take months, if not years, to deliver.

### The challenges of Waterfall

The challenges of Waterfall come from the need to define specifications in detail up-front (when there is often uncertainty about exactly what's needed) as well as the time it takes to get from concept to delivery.

Stakeholders, including customers, don't get to see what's being built until the very end of the process. There's a significant risk that resources get wasted building requirements that were misunderstood, 'gold-plating' (over-engineering what's built), or delivering something that simply doesn't work for the stakeholders.

Also, during the long build phase, customers evolve their thinking, competitors launch products, and the market can change. These change what the new product needs to be. Some requirements may no longer be needed, or new requirements may need to be squeezed in so the product won't fail when launched. It's easy to get things wrong.

### The attraction of Agile

Agile approaches address these challenges by minimizing upfront planning and documentation, shortening the delivery cycle, and

engaging customers (or customer proxies) frequently during the development process. These make it possible to quickly release product versions and validate that they deliver what's needed. If successful, then the initial release can be extended with small iterations. But, if the initial product doesn't fit the market need, then the decision can be made to stop investment or 'pivot' to a better solution. Even if the first product is a failure, the cost to the business is low because of the short release cycle.

The mantra is... if you're going to fail, then **fail fast**, so you can learn quickly and either stop investing or change to something better.

## Is Waterfall dead?

Waterfall or stage-gate processes still have their place. Waterfall may still be the best option when requirements are well defined (such as when building a standard API) or when there are projects with complex and interconnected hardware and software components.

Even in these cases, most companies that use Waterfall are pushing for shorter release cycles. It's not uncommon for companies who previously released every 2-3 years to shorten their cycle to every 6 months – 1 year. This gives some of the benefits of an Agile approach as there is the possibility of releasing smaller, less costly increments to validate that the product meets market needs.

It's also frequently the case that organizations that do Agile development have governance that uses a gated process. This enables senior stakeholders to validate whether planned outcomes are being achieved before committing further funding.

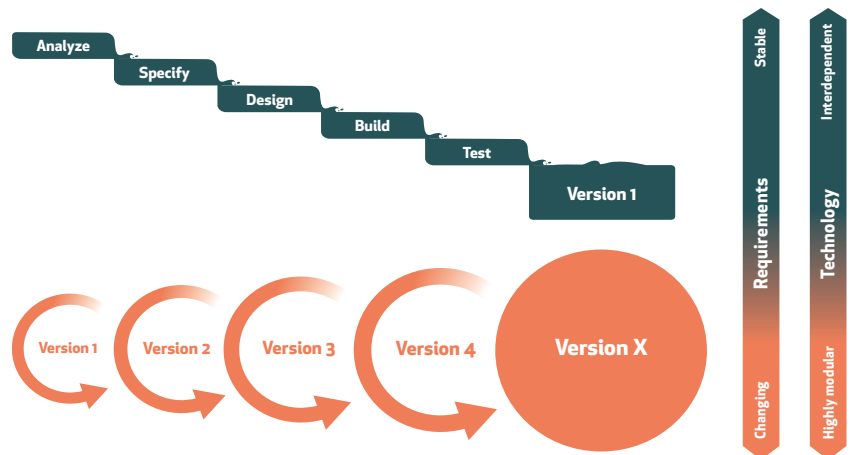


Fig. 1 Agile s. Waterfall development approaches

## The Manifesto for Agile Software Development

In February 2001, seventeen leading software developers got together to discuss an alternative to the heavy, documentation driven software development processes of the time. Their output was the Manifesto for Agile Software Development – <https://agilemanifesto.org>. If you read through the manifesto and particularly the 12 principles that underpin it, you will get a pretty good idea of what Agile software development is all about.

### The Manifesto for Agile Software Development

We are uncovering better ways of developing software by doing it and helping others do it. 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

That is, while there is value in the items on the right, we value the items on the left more.

The 12 principles behind the Agile Manifesto are listed below:

1. Our highest priority is to satisfy the customer through early, continuous delivery of valuable software.
2. Welcome changing requirements, even late in development. Agile processes harness change for the customer's competitive advantage.
3. Deliver working software frequently, from a couple of weeks to a couple of months, with a preference for the shorter timescale.
4. Business people and developers must work together daily throughout the project.
5. Build projects around motivated individuals. Give them the environment and support they need, and trust them to get the job done.
6. The most efficient and effective method of conveying information to and within a development team is face-to-face conversation.
7. Working software is the primary measure of progress.
8. Agile processes promote sustainable development. The sponsors, developers, and users should be able to maintain a constant pace indefinitely.
9. Continuous attention to technical excellence and good design enhances agility.
10. Simplicity – the art of maximizing the amount of work not done – is essential.
11. The best architectures, requirements, and designs emerge from self-organizing teams.
12. At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behavior accordingly.

## Agile adoption

From our 2022 annual survey, we know that 34% of companies use pure Agile, 29% use a hybrid approach (a 2% increase from our 2021 survey), 26% use Agile or Waterfall depending on their product, and 11% use Waterfall only.

## We're Agile but...

When product people are asked about which approach is used, they often say, 'I'm using Scrum, but...'. This is because most correctly customize and adapt Agile practices such as Scrum to fit with their organization's existing processes and the type of projects they are working on (and in some cases, various organizational and behavioral malaises mean that they're really Agile in name only).

The hybrid approaches have colorful names; for example, *ScrumBan* is a mix of Scrum and Kanban. *Scrum-fall*, *WAgile*, or *Water-Scrum-fall* are where Scrum is used during development, but the surrounding business processes for market analysis, business cases, launch, etc., are Waterfall.

A 2020 survey of Agile methods by VersionOne identified the most common Agile approaches – these are shown in the pie-chart on the next page.

They affect, for example, how requirements are identified, prioritized and communicated. The frequency with which what's been built will be shared and approved, as well as the processes for releasing to a live environment ready for customers to use.

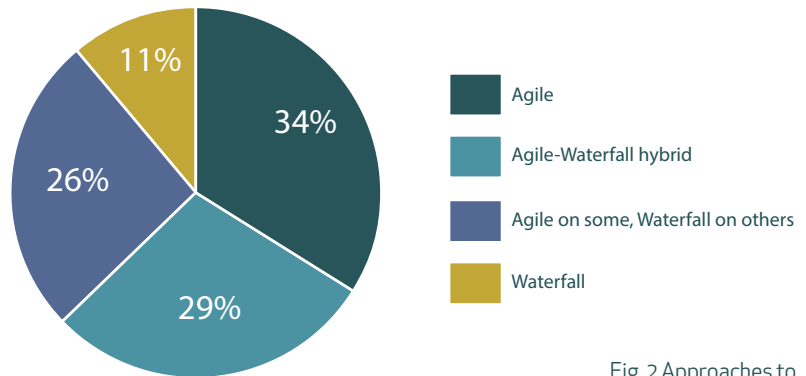


Fig. 2 Approaches to development

"We considered all the Agile methods as a single 'toolbox' from which to select an approach that best suited our context (combining Scrum ceremonies with practices from Kanban, XP, etc.)."

**Paul Gibson,**  
Ordnance Survey

## Making Agile work in a business

Agile software development techniques have been around since the 1970s, and the Agile manifesto was first created in 2001. With this long history, many organizations have grown up using an Agile development approach.

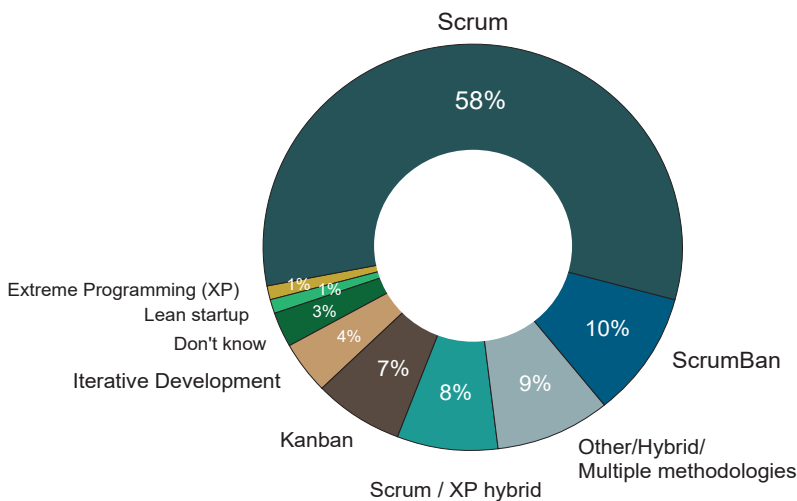


Fig. 3 Source: Agile methodologies. Version One 14th Annual State of Agile Report 2020  
Note that numbers don't add up to 100% because of rounding errors

However, if a company does move from Waterfall to Agile, it is usually initiated by the development team, and they typically go through major re-tooling and process improvements to enable the shift. In either case, to make Agile work, the development team need close collaboration with customers (or customer proxies) to write and prioritize requirements

and to validate that what's built meets the market requirement. This is where Product Owners and Product Managers play their part – managing requirements, signing off on work done, and answering questions about customers.

But, there is a fundamental clash in many businesses. Agile, by its very nature, means adapting to change. It is, therefore, difficult to predict where you'll get to and by when. ►

"Ask anyone to describe Agile, and inevitably they'll use terms like sprint, product backlog, product owner, and sprint planning.

Notice a trend?

Those are all terms specific to Scrum that have become, for better or worse, part of the ubiquitous language of Agile. The reason for that is simple. The Scrum framework has won the market share wars as the most commonly used framework when organizations adopt Agile. That popularity leads many people to conclude that Agile = Scrum. In reality, Scrum is one of many frameworks that you can use as a starting point to approach work in an Agile fashion."

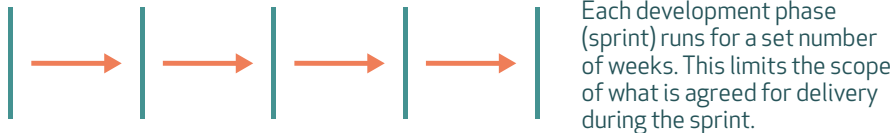
**Kent McDonald, Product Manager & Founder, KBPMedia**



## Agile methods in brief

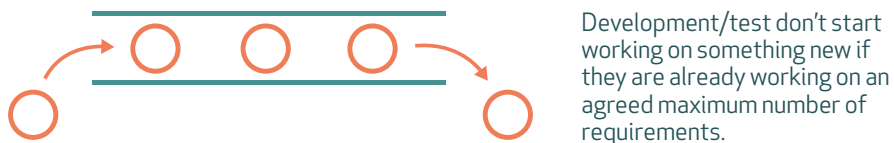
**Scrum** is the most widespread and popular Agile software development approach. It defines key roles and a project delivery framework that delivers short, time-boxed, incremental development releases called sprints (see article on page 11).

### Scrum – time boxed delivery



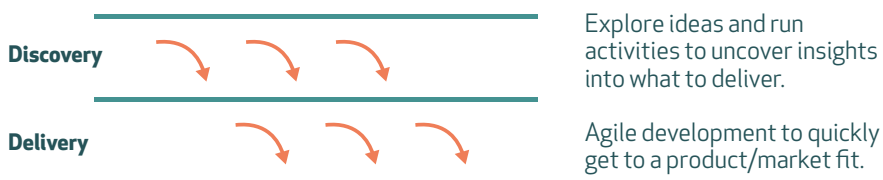
**Kanban** is an alternative to Scrum that optimizes workflow. While Scrum is time-boxed, Kanban limits the amount of work allowed in each stage of a build. For example, imagine a prioritized queue of requirements waiting to go to an *'In development'* stage of build followed by an *'In testing'* stage that validates what's been built. A limited number of requirements are allowed in each stage. Once capacity becomes available, then additional tasks are 'pulled' from the previous stage. There is no time limit on how long each requirement should take to complete - though, in the interests of agility, each will typically be sized so they can be completed within days or weeks.

### Kanban – capacity based delivery



**Dual-track Agile** refers to a way of working where product discovery (see article on page 23) and product delivery are happening at the same time. Discovery is a continual process that identifies opportunities that can then be validated as the "solution space" is explored with customers. Once ideas have been validated through Discovery, they are fed into the Delivery track. So, Discovery fills the work backlog, and Delivery empties it.

### Dual-track Agile – one team, two tracks



# FUNDAMENTALS

In contrast, to run a business, senior management want to know when a product will be ready, how much it will cost and how much it will sell. In addition, there are many other parts of an organization that are critical to creating a successful product. These teams, such as procurement, legal, and marketing, may be doing activities that have a long lead time. To optimize their workflow, they need to know well in advance what's needed and by when.

## Conclusion

Every business wants to be able to do things more quickly and flexibly (to be agile) but implementing an Agile development methodology is only one part of the answer.

Organizations adopt Agile at different levels and in different ways. Some are born using Agile, and some are still experimenting. Some have a small Agile team; others use it for all their development. And a few have fully embraced an Agile approach across the whole company.

If you listen to some advocates, you could be forgiven for thinking that Agile is a 'silver bullet' which can solve all your company's problems. But every business has deadlines, commitments, constraints, and an established company culture. And most businesses have existing products, customers, processes, infrastructure, and teams.

All this context impacts how Agile can work in an organization.

### "Agile vs. agile.

The first 'Agile' has 'A' capitalized. It's something to sell. Processes, ceremonies, and rituals to follow. And if you don't follow the rules, it's seen as heresy or an 'anti-pattern.'

The second 'agile' is an adjective. Being able to move quickly and easily. Experimenting, learning, and evolving. That all makes perfect sense as a product manager and company – but that doesn't always mean 'Agile' is the answer."

**Ian Lunn, Founder, Product Focus**

# Scrum

What it is and how it works

**Scrum is a project management framework for Agile product development.** As the most popular and widely used Agile development approach, it has become synonymous with Agile software development. It does not dictate how everything should be done on a project but leaves many details to the development team. However, it does have a set of defined roles, rituals, terminology, and tools. The following section describes a classic Scrum implementation.

## Scrum explained

Scrum is based on a series of short and focused development projects called sprints. These are time-boxed, which means they must deliver by a set date. The deliverable from each sprint should ideally be something that a customer could use (known as a potentially shippable product increment).



Fig. 4 Scrum is named after the close-knit shoulder-to-shoulder formation used in the sport of rugby

## Roles

There are 3 defined roles in the delivery of Scrum projects, the Product Owner, the Scrum Master, and the team.

**The Product Owner** is the customer proxy and is responsible for ensuring that what the development team build delivers value to customers. Fundamental to their role is a good understanding of customers and their needs. With this insight, they select, prioritize, and communicate requirements, often written as 'stories,' with the development team. The prioritized list of stories is called the *product backlog*, and the process of adjusting priorities and breaking them down is sometimes known as *backlog refinement*. Product Owners normally sign off what has been built to confirm that it is complete

# SCRUM

**The Scrum Master** champions and facilitates the Scrum process coaching the team and tackling any impediments (i.e., blocking issues) that get in the way of delivery.

**The team** is a cross-functional group of, typically, 4-10 people who build software (effort estimation, analysis, design, development, testing, documentation, and progress tracking).

## Process

At the heart of the process is the sprint. This is a set block of time, usually from 2-4 weeks, where the team builds the requirements agreed with the Product Owner.

The **sprint planning meeting** happens immediately before a sprint. It finalizes the list of requirements to be built. In the meeting, the team will discuss the potential requirements, refine their view on the effort required to do them, and agree on what will be delivered.

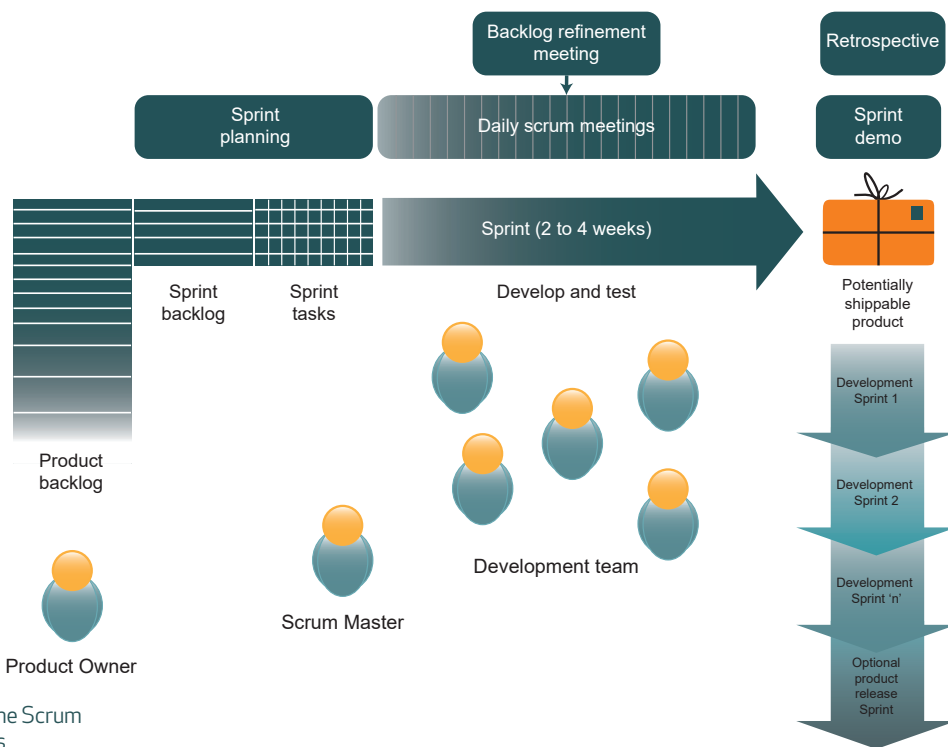


Fig. 5 The Scrum process

During each sprint, the Scrum Master will run a **daily stand-up**. This is a short, e.g., 15-minute review meeting to track what's going on and takes place face-to-face or remotely. Attendees answer the questions. *What have you done since the last meeting? What will you do between now and the next meeting? And Is there anything that will stop you from doing what you have planned?* If issues are raised, then it's the Scrum Master's job to take those issues offline (rather than wasting time during the stand-up) and ensure they are resolved.

It is increasingly common to run a **backlog refinement meeting** halfway through a sprint in preparation for the next sprint. This meeting allows the team to discuss and estimate requirements from the prioritized backlog. Input from the team is key in aligning on priorities. It also helps the Product Owner identify gaps in customer insight and major ambiguities in the requirements, so they can be resolved in readiness for the next sprint planning meeting.

Once the sprint is finished, a **sprint demo** is organized to show the various stakeholders what has been delivered, to get sign-off from the Product Owner, and to provide the team with direct feedback.

Sign-off from the Product Owner means that the requirement is 'done.' It's important that the definition of 'done' is clear and agreed. As a minimum, it will mean that what has been built has passed its acceptance tests and that it incorporates 'standard' requirements (e.g., help screens, alignment to product architecture, and design). A broader definition might require that release notes and user documentation have been created, that the release is packaged and ready to go live, and a record made of any technical debt created. ►

"Possibly the most impactful tool a Product Manager can implement is an agreed prioritization method. So many organizations waste lots of energy arguing about what should be done next; there are always more good ideas than the capacity to implement them, so being able to constantly focus on progress without conflict gives an organization the best chance to succeed."

**Aidan Dunphy,**  
Product Focus  
Senior Consultant,  
Coach & Mentor

## Terminology confusion

Scrum comes with its own terminology, and it's easy to assume that everyone means the same thing when they mention terms such as stand-ups or stories. If you're new, it's always worth checking.

One client recently talked about using 'stories' to describe requirements, and our assumption was that they were using 'user stories' or 'job stories.' In fact, this Agile organization was using classically written requirements, i.e., 'The product shall do XYZ.'

In another organization, while talking to the Chief Product Officer, we asked about the Product Owners' responsibilities in Agile only to be told that they weren't Agile but were wholly Waterfall in their approach to development!

# ESTIMATION



Fig. 6 The Fibonacci sequence is 1, 2, 3, 5, 8, 13, 21 etc. Each number is the sum of the two preceding ones. It's named after Leonardo Fibonacci, a 12th-century Italian mathematician. Image [www.fibonacci.com](http://www.fibonacci.com)

Following this, a sprint **retrospective meeting** is held to review how things went, whether the forecasts for work completion were accurate, and to identify any other issues. The focus of this meeting is on learning, so the next sprint is better (stakeholders outside the core team are not invited). It's an opportunity to put the Agile mindset into practice with experiments on ideas for improvement, for example, running a workshop to better understand a problem area.

## Prioritization techniques

There are various considerations when prioritizing in the sprint planning or backlog refinement meeting. These include the goal for the increment to the product, the value to customers of new functionality, user experience improvements, alignment to strategy, requirements interdependencies, or reducing current or imminent technical debt.

A common approach is to evaluate requirements based on their value (to the organization) vs. cost and urgency. One example is Weighted Short Job First (WSJF), described in the table below.

<b>Weighted Short Job First (WSJF)</b> This prioritization technique prioritizes the delivery of high-value requirements. It uses two attributes of each requirement: <b>Value</b> and <b>Duration</b> (to build). Value will often be measured by a <b>Cost of Delay</b> or a <b>Fibonacci number</b> . Duration is measured in days or weeks of effort or a Fibonacci number. Dividing Value by Duration gives a priority order with the highest scoring items done first.		Value (Cost of Delay)	Duration (weeks)	WSJF score	Priority
	Feature A	£10,000 / week	4 weeks	2,500	2
	Feature B	£30,000 / week	20 weeks	1,500	3
	Feature C	£30,000 / week	2 weeks	15,000	1
		Value (Fibonacci)	Duration (Fibonacci)	WSJF score	Priority
	Feature 1	8	3	2.7	2
	Feature 2	21	21	1	3
	Feature 3	21	2	10.5	1

## Common variations to the 'standard'

While this article shows the standard way of working, there are many variations made to ensure it works in a particular business.

For example, while purists argue the team should be self-organized, in practice, we've seen the need for experienced members to provide insight into estimation and work allocation.

## Planning Poker

A challenge with prioritization is the estimation of effort and value. It can be tough to estimate value in monetary terms, and it can be tough to estimate in working days of development effort. As an alternative, value and effort for each story are often measured relative to each other using a somewhat abstract metric such as t-shirt sizes, e.g., Extra small, Small, Medium, Large, Extra Large, or points, e.g., 13 points of effort.

With points of effort, to avoid endless debates over whether a story is estimated at one of two close numbers, e.g., 17 or 18 points, the team are asked to choose an estimate from a Fibonacci sequence (1, 2, 3, 5, 8, 13, etc.) where there are fewer big numbers. All estimates are revealed at the same time (so as not to influence each other). Significant variations are discussed, and a consensus agreed. Any requirement that is given a very high score (e.g., that's likely to take longer than half the period of the sprint to deliver) will be broken down further, and each part estimated and prioritized. This technique is called Planning Poker.



Another challenge is that while each development team is supposed to be multi-disciplinary, we often find that specialist skills are needed, e.g., on databases, on user interfaces, or on security. We see two common solutions to this. The first is to schedule these specialist requirements into the sprint team that has the expertise to deliver. The second is for an expert to be brought into an existing team for a period of time to do what's needed. Either approach creates the challenge of having to plan time with the specialist that has the right skills and experience.

While the result of each sprint might be a 'potentially' shippable product, what often happens differs from this. The sprint itself might be an 'investigative spike' where the development team explores technical approaches to building something rather than doing the build. The team might be operating in a DevOps continual delivery mode where code is automatically tested and, if the business wishes, released directly onto an operational

## What is DevOps?

Tight interworking between the people in Development and Operations groups to enable the faster release of newly built software into production. It is facilitated by having aligned development and operational environments, automated testing, near real-time performance monitoring, and other techniques.



# PRODUCT OWNER

platform. A third common outcome is that what's been built is insufficient value to customers to justify either the effort of launch or for customers to upgrade. In this case, multiple sprints will be used to add more functionality, ending with a hardening sprint that performance tests the product prior to launch. This sequence of sprints delivers a higher value product to customers.

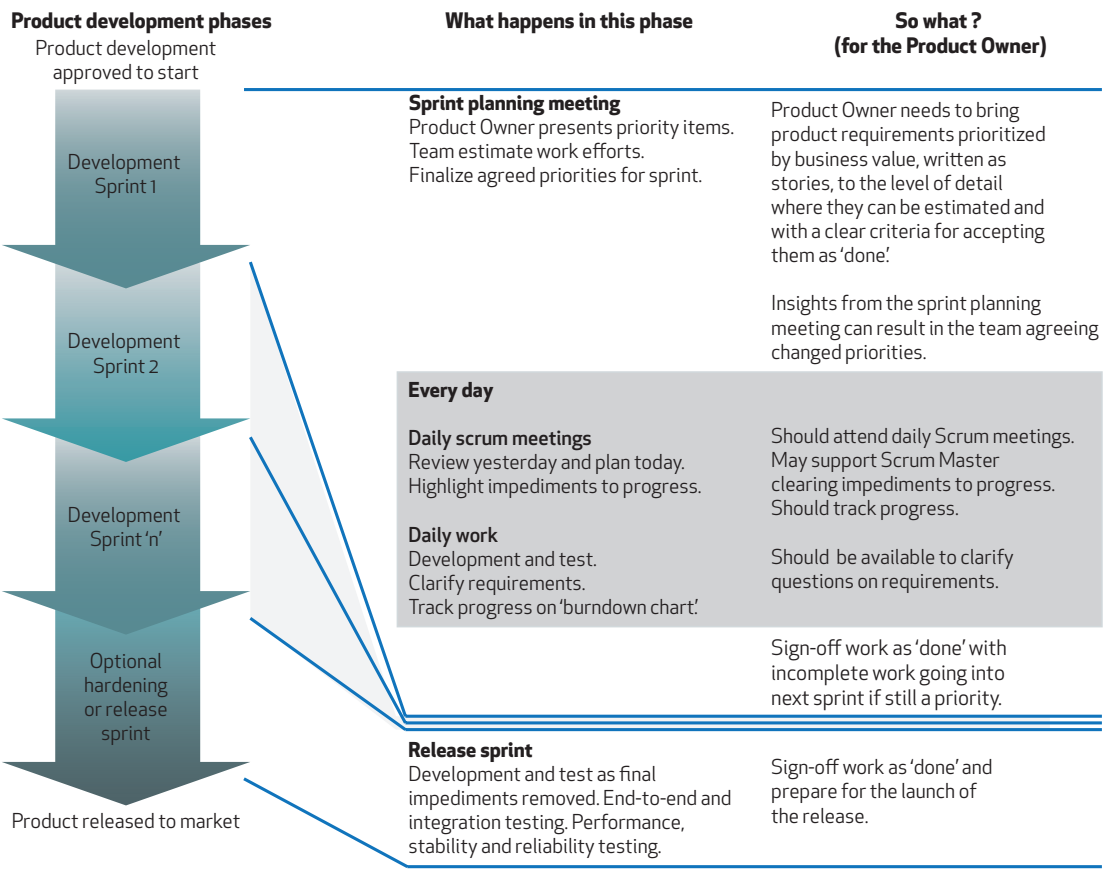


Fig. 7 The Product Owner and Scrum

## The Product Owner and Scrum

A key role for the Product Owner is to present the market priorities for each sprint. This is done by constantly managing the product backlog and prioritizing based on customer insight, development estimates, and alignment to business priorities.



What finally gets agreed for inclusion in the sprint is often a combination of these requirements, input from the technical team on what they must get done (e.g., tech debt, interdependencies, architectural changes), and input from the user experience team.

## Writing requirements

Requirements are documented as stories (typically user stories or job stories) at varying degrees of detail depending on their priority. To start with, requirements are normally documented at a high level, termed an **Epic**. This is a large requirement that describes broad functional needs.

In preparation for the sprint planning meeting the highest priority requirements, those to be built in the next sprint or two that are part of an epic are broken down into smaller requirements (i.e., those that could be built with just a few days effort). The format of requirements is usually that of a user story or job story.

User stories are used where the requirement is to deliver functionality that lets someone do something, and the format is a short, simple description written from the perspective of that person. A typical format is:

**As a** <type of user>, **I want to** <some goal> **so that** <some reason>.

For example: **As a** user of email, **I want to** change my password **so that** I can maintain the security of my account.

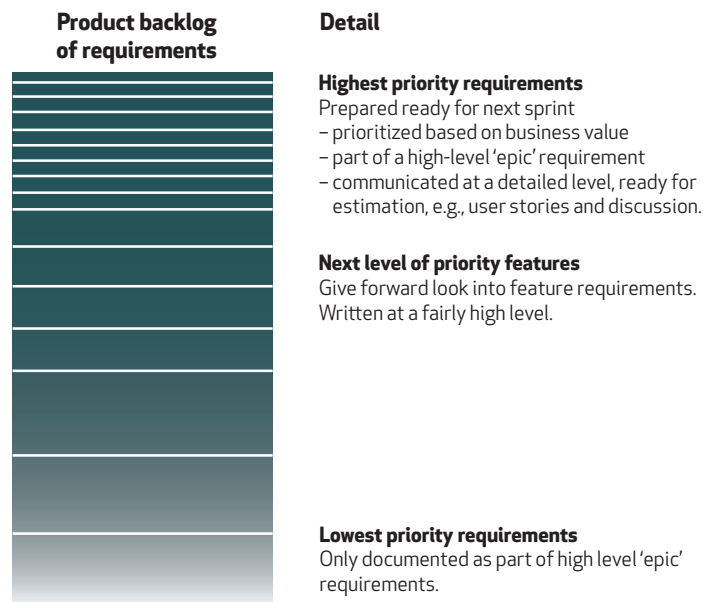


Fig.8 The product backlog

# BURNDOWN

Job stories are used when a piece of functionality is triggered by a system event. A typical format is:

“Agile gives Product Managers great opportunities to offer early-adopter versions to customers and get good, instant feedback – prototyping and user validation help make great products.”  
**Derek Britton, Product management leader**

**When** <trigger / situation> **I want to** <motivations / forces> **so I can** <expected outcome>.

For example, **When** a mail attachment size exceeds a system threshold level then, **I want to** record details of the attachment size and format them into a log, **so I can** get a report of when and how often it happens.

Whatever format is used, acceptance criteria also form part of the story. Often these are started by the Product Owner and refined with the team.

Stories are described in more detail in our **Requirements Product Management Journal**.

## Tracking and estimation

One of the most common things used in Scrum to manage and track

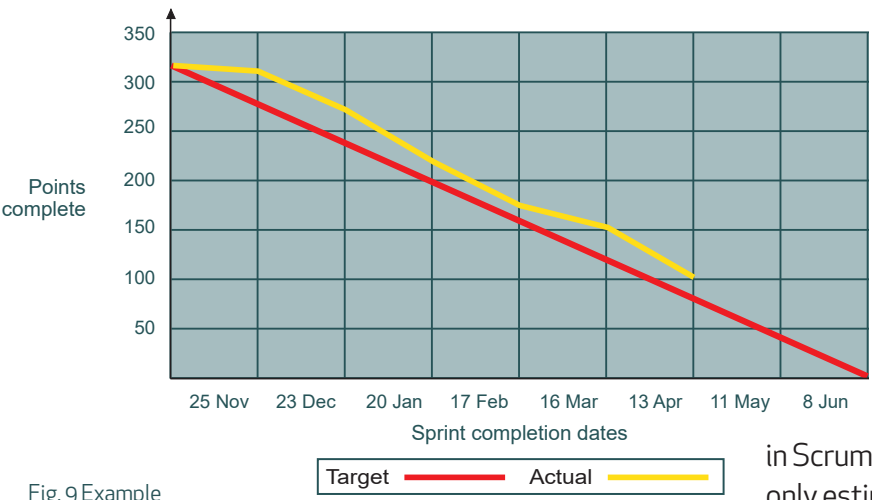


Fig. 9 Example release burndown chart with 4-week sprints

the status of work in a Sprint is a task board (see photo on page 39). Typically, a row is allocated to each user story with columns representing To do, In development, In test, and Done.

As we’ve already described, estimation

in Scrum is often simplified by only estimating development tasks relative to each other, and

this is done by awarding points for the size of each task. As each sprint is the same length of time, the team learns how many total points can be delivered in a single sprint. This is known as the team’s velocity. The velocity of development is tracked using a burndown chart. A burndown

chart records user stories completed and shows the team if they are on track to deliver the volume of work previously forecast. What tends to happen is that velocity will be slow in a new team working on a new product domain. As they build familiarity, their velocity will improve (as will their ability to estimate reliably).

## Great responsibility

The Product Owner sets the development priorities through the product backlog and can change their mind on these priorities at any time up to the start of the sprint when they are to be built.

This gives the flexibility to respond to market and competitor moves or as the understanding of what's needed evolves. Which is great. However, it also means there is a heavy responsibility on the Product Owner to get it right, which they can only do if they spend the time to get the market and customer insights needed.

It's easy for Product Owners to fall into the trap of **'feeding the development machine.'** That means focusing only on writing requirements to make sure the development team are kept busy, and so everyone can point to how much has been delivered. The team becomes a 'feature factory' where success is measured by the number of features built, with little regard to the value of what has been created. The solution is to make sure there is a focus on the business outcome of what's being delivered (e.g., customer retention, customer satisfaction or product sales).

"Product Owner is a role you play on a Scrum team. Product Manager is the job."

**Melissa Perri,**  
Speaker,  
consultant, and  
teacher on product  
management

## Product Owners and Product Managers

Many Product Managers do the Product Owner role. Lots of Product Owner roles expand to take on product management activities. And in other businesses, there are Product Managers and Product Owners working closely together.

However, if you take the Scrum definition, a Product Owner is the role you play as part of the team that is developing the product. And, it is generally accepted that a Product Manager's job is a much wider role, one that is responsible for the overall success of the product. We explore who does what in the next article.

# Focus

Survive and prosper with Agile

If you're a Product Manager wondering how to get the most from Agile and Scrum, we've compiled this list of 5 key areas to help you understand what to focus on.

1. Product Manager vs. Product Owner?

One of the key aspects of Agile is that the development team must work closely with someone who understands what the customers want for the product. The role is critical to drive what is developed. In Scrum, this role is called the Product Owner and involves prioritizing the backlog of requirements, answering day-to-day questions, providing signoffs and feedback. Done properly, it takes a large commitment of effort and time.

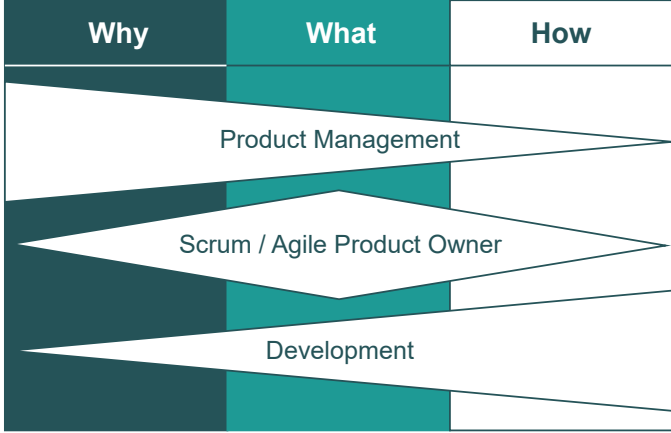


Fig. 10 Adapted from Agile roles framework created by Paul Inness, using the Golden Circle by Simon Sinek

As a Product Manager, you may do the Product Owner role, you may work with a Product Owner, or you may be a Product Owner who also does product management activities.

One way to separate the different roles is to look at their focus on requirements, **Why**, **What**, and **How**. The Product Manager will be concerned to a large degree with **Why** a requirement is important based, for example, on its fit with strategy, value

to customers, value to the business, and competitive positioning. The Product Owner will work on the details of **What** is required and will to a lesser degree, worry about why each requirement is important and how it will be built. The development team need business and market context, but most of their focus will be on **How** to build the requirement and then getting on with it.

The reality is that some businesses only have the Product Owner or the Product Manager role. But if both roles exist, they must be aligned on who does what and the priorities for the product. Typically the Product Manager has the high level view on strategy and epic level requirements, while the Product Owner works on the detailed requirements and engagement with development. To do this effectively, both need to understand customers and users.

“For Product Managers & Product Owners to work in harmony, the Product Manager determines the right products to build, and Product Owner works with Development to make sure they’re built right.”

**Paul Inness,**  
Innovation  
Consultant, Coach,  
& Mentor

## 2. Know your market and set expectations

Development may be capable of updating products after every sprint, but there are markets where product updates carry significant work, major risk, or regulatory implications for customers. In markets like these, such as pharmaceuticals, banking, or telecoms, customers won’t be able to accept new product updates frequently. Instead, they need to test products and train staff before starting to use a new release.

Even if customers can accept frequent releases, a program of re-education will be needed if they’ve become used to seeing long-term, detailed product roadmaps from their vendors. They need to be sold on the flexibility Agile brings, which also means acknowledging uncertainty over the detail of what they will get in each release.

The trick is the careful management of what’s being delivered and communicated to your customers. You need to control what is released, when to release it and when to talk about it.

## 3. Manage internal stakeholders

Our internal stakeholders, just like our customers, value certainty – so they know what’s coming and can plan accordingly. While Waterfall development approaches try to achieve certainty in roadmap delivery, the reality, even with significant resources committed to planning, is that this is very difficult to achieve. Agile recognizes this, and rather than the illusion of certainty, it offers the promise of flexibility.

It’s key that senior stakeholders and the other teams around the organization who are critical to product success (such as marketing, support, and operations) recognize this.

# BEST PRACTICE

"In my experience, one of the biggest challenges implementing Agile is the cultural change. It's not that hard to change processes, tools, or even skills – but changing hearts and minds can be a huge challenge."

**Derek Britton,**  
Product  
management  
leader

While there is uncertainty, it is still possible to create and communicate roadmaps and plans. One option is to focus less on features (i.e., we build this functionality) and more on the outcomes we've enabled for customers (e.g., shorten browsing time for people choosing a movie by 20% – for the Product Manager of a streaming movie provider). As with customers, there is a continual education process needed to sell people on the value of being flexible and accepting of the compromise of not knowing precisely what will be delivered.

## 4. Find customers to work with

The beauty of Agile is how changes in product development direction can be easily accommodated by re-prioritizing the product backlog for the next iteration. To take advantage of this, you need feedback from customers and users to check you are on track. If at all possible, you should be setting up an early-adopter program to recruit customers who are keen to get their hands on new features in return for providing feedback.

It can be a challenge finding customers to work with, so be careful to make sure the customers you choose are representative of the market.

## 5. Build credibility with development

When you are in daily contact with the development team, you are under the spotlight and so need to build your credibility and earn their respect. Development needs to have faith that you understand the market, have thought through the product direction, and can lead the prioritization of requirements; otherwise, they might just go off and do their own thing.

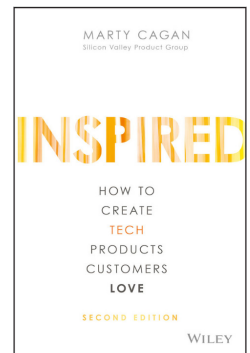
So, make sure you clearly communicate your vision for the product and why things are important. Earn trust by responding to any questions in a timely fashion. And build rapport by learning about their development challenges.



# Discovery

Market Insight and empathy

**The term ‘discovery’ covers a range of techniques that help product and marketing people get the insights they need to improve their products.** While the word has been around for millennia, in the context of products, it was probably first coined by Marty Cagan in his book, *Inspired*. His choice was made, as he says, “...to illustrate the need for an open mind to know what they [Product Managers and teams] can’t know and admit what they don’t know.”



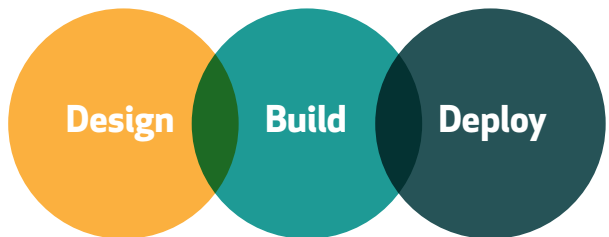
## What's it for?

Often, we find the term used only in the context of trying to uncover breakthrough insights that can spawn great new product innovations. Although that's a key application, the techniques are also valuable in working out where the product team should focus their efforts for incremental improvements of existing products.

It's all about finding nuggets of insight into customer pain points and what they value. Knowing these means, we can better focus our company's limited resources on the areas that matter to customers and align with our business objectives.

These insights influence how we design, build and deploy products.

Where's the source of insight? It starts with getting close to your customers, having empathy for them, getting immersed in their daily life, and keeping an open mind about what they do, the environment in which they operate, and why they make the choices they make.



# BEST PRACTICE

## Guiding principles for product discovery

Teresa Torres, Product Discovery Coach and Founder of [ProductTalk.org](https://ProductTalk.org) has done a great job in her blog on product discovery. She identifies 6 guiding principles that help explore the problem space. A short précis is provided below, but we would encourage you to read the full blog post [here](#). The 6 principles are:

1. Start with empathy for your audience
2. Explore the problem space indefinitely
3. Map your way to clarity
4. Use theory as inspiration
5. Co-create solutions that meet the unique needs of your audience
6. Surface and test underlying assumptions

### 1. Start with empathy for your audience

The start point for discovery is empathizing with the viewpoint of your audience. It can be tough to avoid projecting our own perceptions and making assumptions. What's needed are unbiased insights into their motivations and context to understand what works for them and what doesn't.

### 2. Explore the problem space indefinitely

If discovery is treated as a one-off event that explores the problem space and then moves onto the phase of generating solutions, then it is easy to miss changes and nuances. Also, you'll lack the mechanism to go back, to dive-deeper and re-clarify. There's always more to learn, and that's why adopting continuous discovery practices is so important.

### 3. Map your way to clarity

Our brains are exceptionally good at spatial reasoning and so creating visuals of what has been learned helps unlock powerful insights. Maps come in all shapes and sizes: customer journey maps (that illustrate their journey to achieve a particular outcome), process maps, story maps, network maps, and so on.

"Discovery: work that explores, evaluates, and confirms product options for potential delivery  
Delivery: work that transforms one or more allocated candidate solutions into a releasable portion or version of the product."  
**Gottesdiener and Gorman definition from their book Design to Deliver**



## 4. Use theory as inspiration

It's easy to fall into the trap of creating a solution that's no better than what already existed. Don't re-invent the wheel - innovate in places where better solutions are needed.

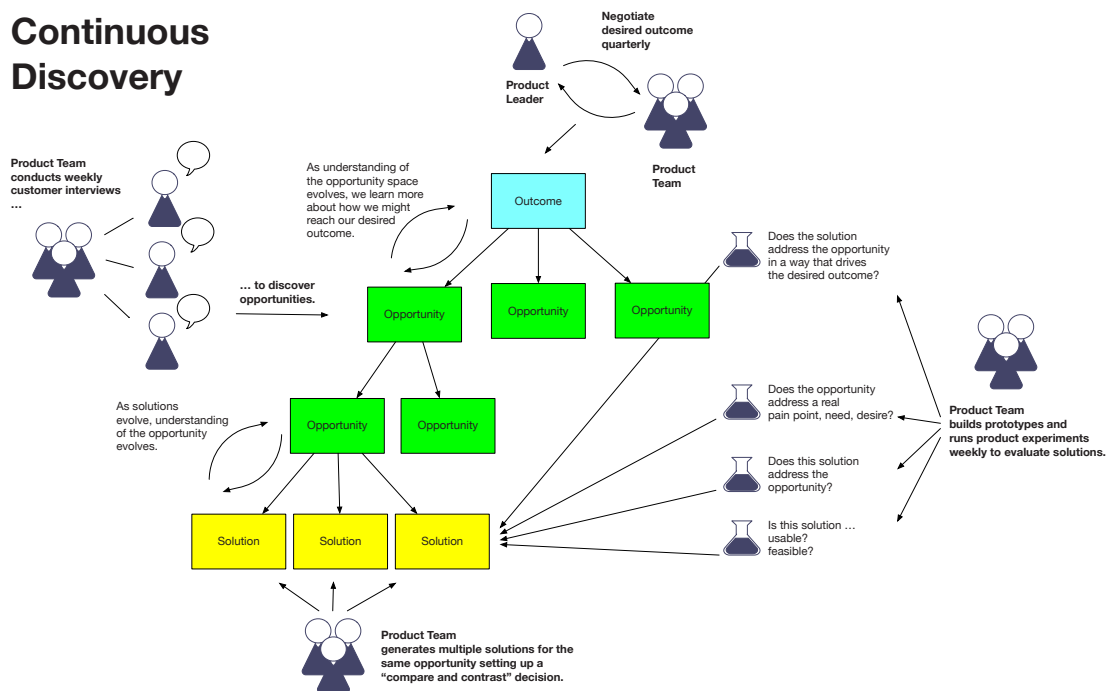
To avoid this, use theory and first principles to help frame the problems we see. This means considering human truths (e.g., psychology, biases, sociology) that are pretty stable over time. It also means considering technology truths (e.g., technology preferences for mobile vs. desktop) that may change quickly.

## 5. Co-create solutions that meet the unique needs of your audience

When we hear a problem, too often we jump to the first solution that comes to mind, and over time, we fall in love with our ideas. We start failing to ask, "How is this situation different?" and what might we learn to ensure our solutions address customers [market] specific needs.

Fig. 11 Opportunity solution tree

### Continuous Discovery



PRODUCT TALK

© Copyright Product Talk LLC [www.ProductTalk.org](http://www.ProductTalk.org)

# DESIGN THINKING

Teresa advocates using an opportunity solution tree, shown below to visually map solution ideas to the opportunities found in discovery interviews as a way to guarantee that real needs are being addressed.

## 6. Surface and test underlying assumptions

After we've come up with a product idea, it is tempting to leap into testing it out with a prototype or Minimum Viable Product, but it's good to pause and reflect - product graveyards are littered with excellent technology that nobody wanted.

It will save you time and energy when you get to development if you are able to validate the assumptions behind your product idea.

For example, you might use story mapping to identify key actions that you anticipate people will take to satisfy their needs. Each action is based on an assumption – what will cause these people (actors) to take those actions? Is it realistic that they will?

Also, the success of any product depends on its desirability to users (as well as many other factors). Validating your assumptions on why your product is desirable can help you avoid adding to the product graveyard!

## Techniques to help product discovery

Teresa's post emphasizes the importance of ongoing exploration of the market problem space. There are a multitude of tools and approaches that may help do this, and we've picked out three of these below.

**Design thinking** is used to explore customers context and challenges, **Jobs To Be Done** helps understand and prioritize what's important, and **Design Sprints** help to create and learn from a prototype in just one week.

## Design thinking

Design thinking is focused on creating empathy with a customer... getting under their skin to really understand their situation, what they're trying to achieve, how they go about it, and the problems they face. It requires using techniques such as customer interviews, 'day in the life of a...', and observations. During these sessions, there's a need to work hard at undoing our in-built biases, preconceptions, and

"The time spent on discovery should be relative to the complexity or the impact of the problem you and the team are attempting to solve.

If the challenge is a relatively simple change, the discovery process can be relatively short, including techniques such as Benchmarking, Prototyping, and User Testing.

If the team is tackling a very complex challenge involving multiple stakeholders, then it is worth investing more time in discovery using methodologies like a design sprint and extensive user studies.

There is often pressure to rush the discovery process and deliver things quickly. However, teams should focus on delivering the right thing to their users, and the product manager should be an advocate of this goal."

**Yoav Farbey, Lead Product Manager, Business Hub**

assumptions to be open and continually re-frame the problem space based on what we're discovering.

The results are a set of unstructured observations that later need to be followed up with discussions to explore the emotional and functional needs of customers.

This product discovery is messy, and the output needs refining. The outcome should not be a set of fluffy, vague statements about what customers value but should identify the sweet spot of ideas that are: **Desirable** (to the customer), **Viable** (it makes sense for our business to build a solution), and **Feasible** (technically we can do it).

All sounds fine, but who do you talk to? A challenge we've found is product teams talk to the customers with whom they have an existing relationship – it's the easy choice. Unfortunately, that can also mean they find the answers that they already knew. So, it's good to work with some of the outliers, the prospects or customers that behave in a different way to the norm – these might give insights that you won't get from the mainstream. While these need validating as representing a target market segment, having this different perspective can be great at getting ourselves out of a rut.

Doing all this work takes time and money. If speed to market is important (as it is in most markets), then it can be tricky getting a balance between how much time you spend on this research to get insights and getting going on development. What you're aiming to do is to get enough evidence to be able to make decisions on what you think customers will value, build the minimum that will deliver this, and iterate from there while continuing your process of discovery.

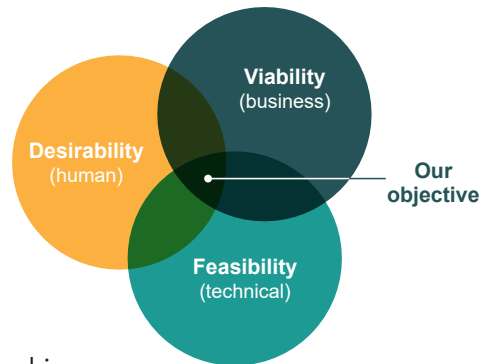


Fig. 12 Design thinking

## Jobs to be done

Jobs to be done is another technique that's really useful at getting under the skin of customers. Its purpose is to identify the worthwhile opportunities for delivering things your customers value.

# DESIGN SPRINTS

It starts with getting insight into what people are trying to achieve, e.g., the outcomes they are striving for (such as a customer support team reducing their cost per call or getting feedback to tell them that they're doing a good job). Each of these outcomes is then assessed to work out where there is an opportunity for improvement and how these might be valued by customers.

The research is based on 2-phases of analysis. It starts with in-depth research based on surveys or, ideally, one-on-one meetings. This digs into the reality for people in particular roles... their context, decisions,

behaviors, and solutions (the outcomes of 'what' they are doing) and discussions on the 'why' behind what's been observed.

The second phase is a survey of more people in similar roles, so there is sufficient data to be able to draw reliable conclusions. The basic output of a Jobs To Be Done research is a graph like fig. 13. Each diamond represents a particular outcome, and the axes of importance and satisfaction (with current ways of working) help categorize each job as being over-served, adequately served, or under-served. The underserved segments are those that are great opportunities for

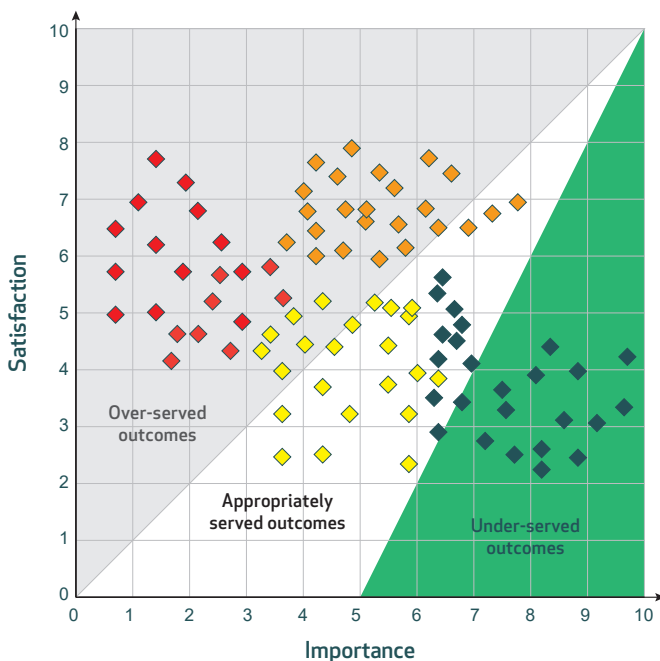


Fig. 13 Jobs to be done

innovation, while the other categories point to opportunities for incremental improvement in a product.

The challenge with this technique is that it takes time, expertise, and access to customers to do. So, many companies use outside agencies to run this research for them.

## Design sprints

The term Design Sprints comes from Google Ventures, and a playbook is provided in the book, *Sprint*. However, it is an approach that

has worked well in many other companies – bottomless wallets are not a prerequisite!

A design sprint typically lasts a week though we've seen them happen over different periods from 4 days to several weeks. It involves a team of people familiar with a particular product area that come from diverse disciplines, e.g., product management, Product Owners, development, customer experience, marketing... and customers willing to assess what's been created during the design sprint.

We like the approach because it provides customer validation (or not!) of the riskiest assumption for a product in just a week.

There are many positive aspects of running a design sprint. None more than the Test day. We like this for two important reasons. First, you get real customers to test and get their feedback really quickly. The whole team sees this live, which makes the direct feedback really powerful. Second, you only need 5 customers to get to around 85% assurance of the feedback. So, with just 5 customers, in a week, you can determine if you've missed the target, discover what really matters and find out what gets people excited.

Just one last comment – Design Sprints may only take a week to run... but they take good preparation to be worthwhile – and follow-up to build on what's been learned!

## Conclusion

Good product management is working out the right product for the business, building the product right, and helping the business to sell it. Discovery helps across all these aspects:

- Creating empathy for the customer and using these insights to drive your innovation and marketing.
- Making product decisions – what to drop, what to keep, and what to focus on.
- Being more creative during the development process and maintaining the interest and motivation of the development teams.
- Improving your market engagement by linking the customer outcomes enabled by your product to the benefits that will be talked about by Sales and Marketing.



## Riskiest Assumption test (RAT)

One technique we've seen is to decide on the riskiest assumption you've made for your product idea and then focus on trying to validate it.

# Agile has to scale

....Get used to it!

**When you're not a startup working in a garage (or never were).**

Let's begin at the start-up end of the world. Small software companies are often built from the ground up using Agile development approaches. Also, being small, start-ups usually have the agility to quickly adapt their way of working while still keeping everyone aligned

- the whole organization is Agile in the broadest sense of the word.

This works well in a small team of close-knit people, sharing a common goal of delivering great products that their customers really value.

But, how does that same organization keep being Agile when it grows to a large number of people, with mature processes, complex products, perhaps operating in multinational markets?

Also, what about the more common scenario of existing, mature organizations looking to evolve and become more Agile – what do they typically

do, and more to the point, what should they do?

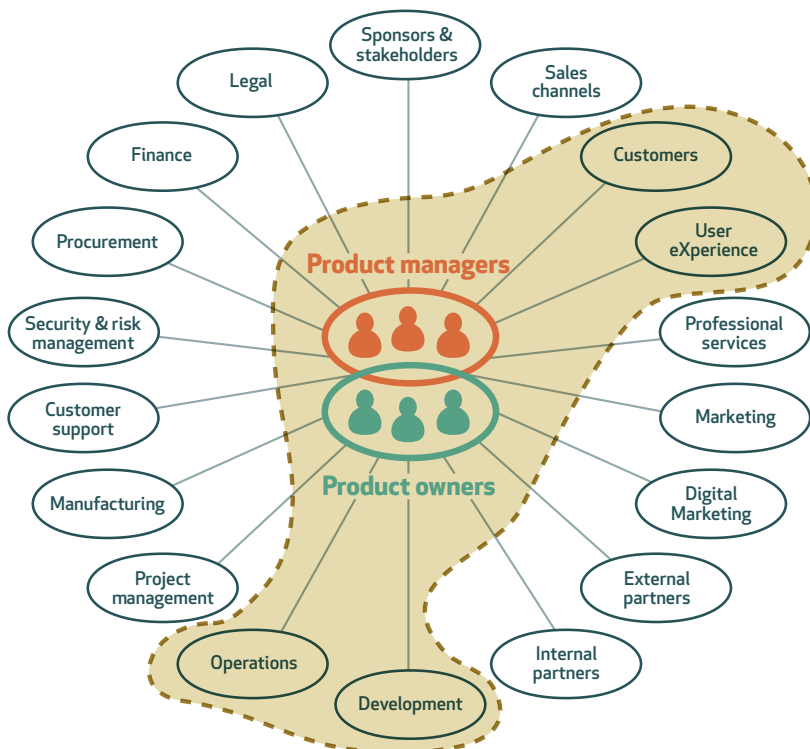


Fig. 14 Areas typically most impacted by small scale Agile implementations

## How does Agile happen?

In many mature businesses, Agile techniques are first introduced by Development, keen to improve the value of what they deliver or frustrated at having to build requirements that are poorly prioritized or unsupported by good rationale or market context.

The areas impacted by the introduction of Agile aren't just Development but are typically Product Management, User eXperience, and Operations as well as customers.

In large organizations, we'll often hear of pockets of excellence. *"This team has some fantastic approaches to getting customer insight"* or *"That team has achieved great success with its new products."* These teams are often in an 'innovation' center or have achieved some other independence from central control, which has been key to their success.

Limiting Agile to development teams or small pockets of excellence means that some of the Agile challenges (e.g., working with other teams in the organization; governance and senior stakeholder engagement; aligning expectations with sales teams, marketing, and customers) can be ignored or worked around. However, it also limits the value that Agile can bring to an organization.

"If, like me, you've ever been in a standup with 19 attendees, you'll know that adding people to a big team slows it down, rather than speeding it up; you reach a point where there are simply too many people to collaborate effectively. Teams need to be small and work best when empowered, but it's crucial to have a clear product strategy to ensure they don't work against each other."

**Aidan Dunphy,**  
Product Focus  
Senior Consultant,  
Coach & Mentor

## Scaling Agile and Agile transformation

The first level of scaling is moving from a single team using Agile (or small pockets of excellence) to the whole of Development. The next level is the broad adoption of DevOps (see explanation on page 15) to enable the fast deployment of newly built software into production. The final level of scaling is known as Agile transformation and is when a much broader range of teams in the organization aim to become Agile in their mentality and way of working.

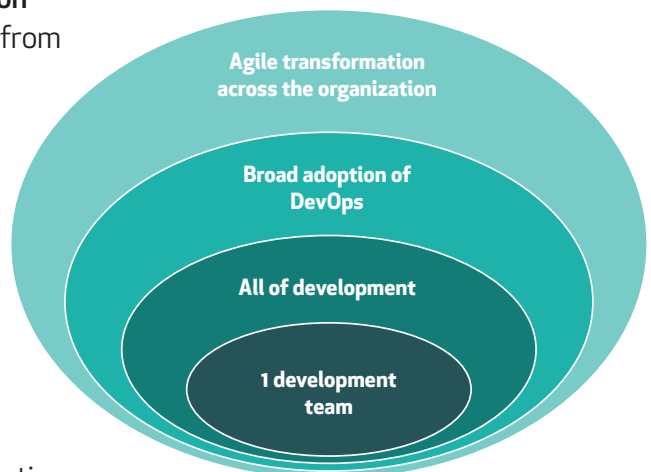


Fig. 15 Scaling Agile

# BLOCKERS

So, what are the challenges when scaling Agile?

## Challenges in scaling

There are many approaches to scaling Agile. Each has their proponents, and all are at different stages of maturity and adoption.

The VersionOne 14th Annual State of Agile report, released in May 2020, provides useful insights from those companies that have implemented Agile approaches to development.

That report identifies the top 3 blockers to achieving success with Agile, which are: general organizational resistance to change, not enough leadership participation, and inconsistent processes and practices across teams.

We've often seen organizations that claim to be 'Agile' but that have inflexible business processes, overbearing governance, or micro-management. They seek certainty rather than welcoming experimentation and allowing failure. This is sometimes known as 'Agile-theatre', where the rituals and ceremonies of Agile are followed religiously, but the business mindset is just not Agile.

## Approaches to scaling

When it comes to scaling Agile, SAgile is by far the most popular

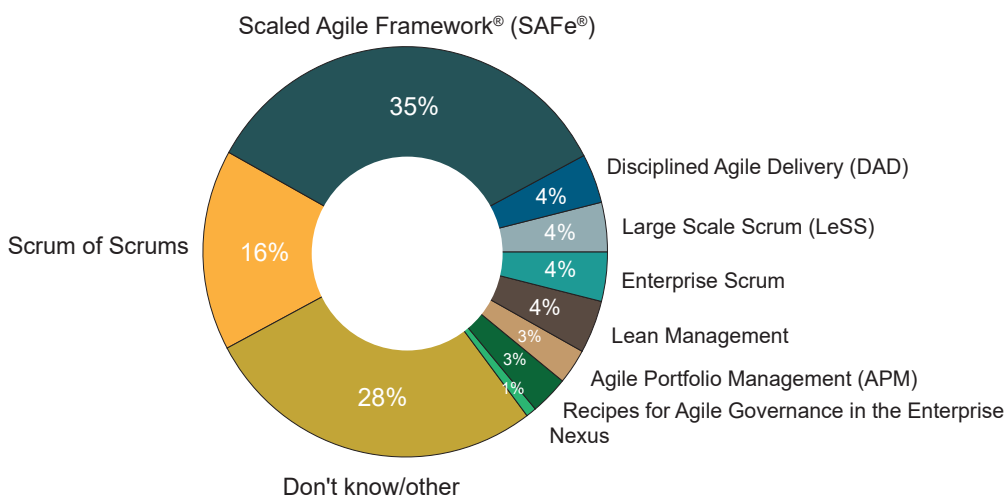


Fig. 16 Source: Scaling Methods and approaches. VersionOne 14th Annual State of Agile Report 2020  
Note that numbers don't add up to 100% because of rounding errors



approach, with 35% of respondents citing it as the one used in their company.

The VersionOne report identifies that while agility is still largely confined to development, IT, and operations, there is momentum building around broadening this to achieve business agility. This is reflected in a recent update to SAFe 5.0.

There has been considerable work over the past decades to highlight and champion Agile approaches to larger organizations.

For those unfamiliar with Agile scaling, the different approaches such as SAFe, LeSS, DAD and DSDM can easily come across as a long list of acronyms with little to differentiate them.

There is a wealth of information about each available online – we show below what we’ve learned and the high-level implications on Product Managers.

Included below is a brief synopsis of those approaches we feel are most appropriate to more complex, larger organizations where scaling is a particular challenge.

## SAFe

Scaled Agile Framework for enterprises is the most mature and widespread framework for large organizations. While it is customizable (and there are coaches that can do this), each implementation will be relatively prescriptive on how things should be set up.

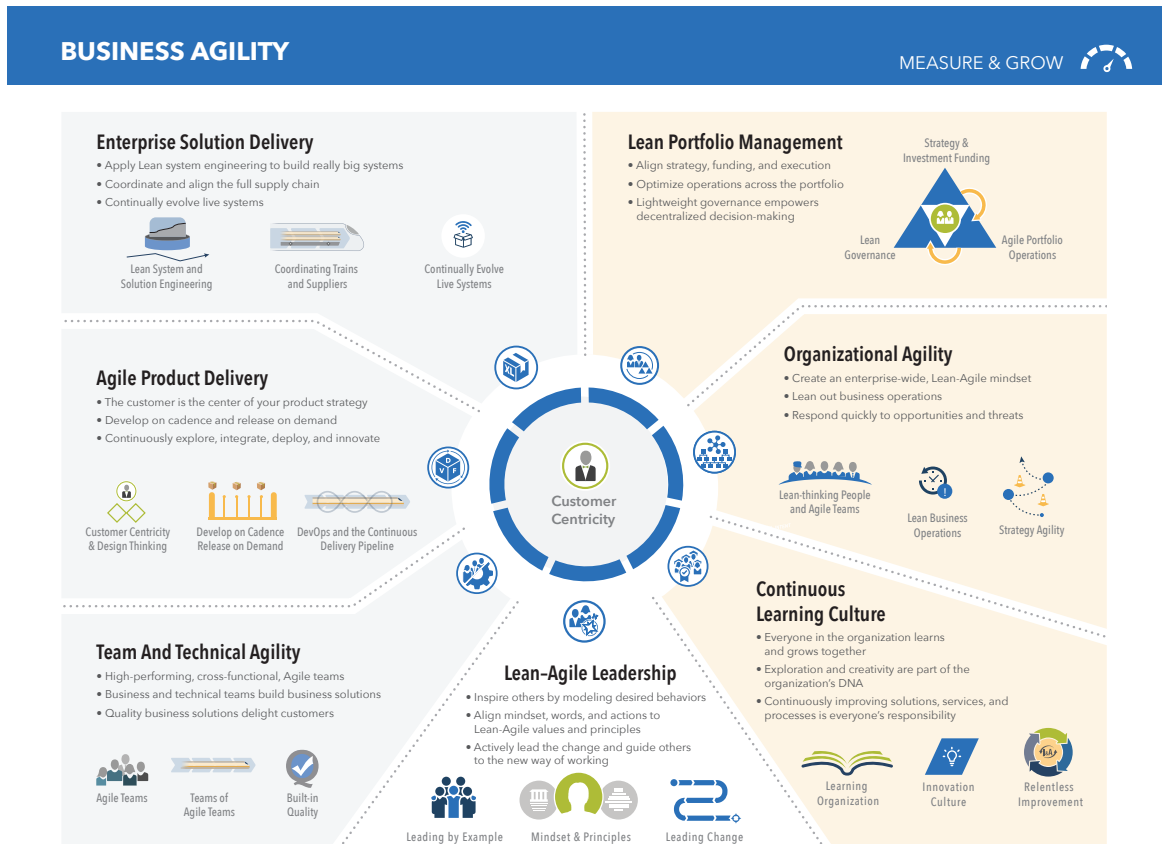
It is suitable for both software and those companies building software/hardware products. There’s undoubtedly an appeal in the simplicity of standardization “apply SAFe 5.0, and we’re done”, but the reality is that companies’ contexts are all different, and customizing the detailed implementation is normally needed.

The Product Manager has a key role in SAFe implementations. They define the product strategy and use market and business insight to prioritize and ensure products both “delight customers and deliver economic value to the business.” In some implementations of SAFe, the Product Manager’s traditional responsibilities may be split with a Solution Manager; in others, the Product Manager takes on a combined role. ►

“Agile truly works when there is a holistic approach across the entire organization. This needs to be from the CEO across to every team, including, of course, the development teams. By this, I mean, the whole company needs to work with the understanding and adoption of the Agile ways of working.”

**Yoav Farbey, Lead Product Manager, Business Hub**

# SCALING AGILE



**SAFe** | PROVIDED BY **SCALED AGILE**

Fig. 17 SAFe 5.0 Overview. Source [www.scaledagileframework.com](http://www.scaledagileframework.com)

## Scrum of Scrums

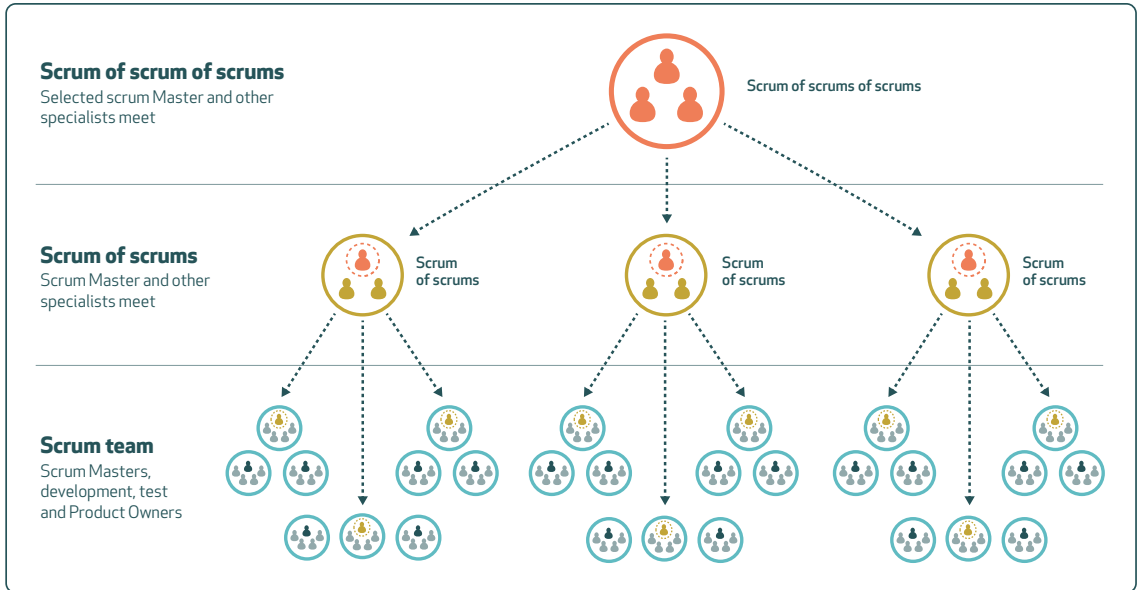
One of the earliest approaches used to co-ordinate the work of multiple Agile development teams was Scrum of Scrums. It is still widely used, with 16% of the VersionOne survey using it as their approach to scaling.

The Scrum of Scrums approach supports cross-team collaboration and planning highlights dependencies and addresses points of integration or conflict.

Scrum of Scrum meetings are run as regular time-boxed meetings of

representatives from multiple scrum teams. They typically follow each teams' daily stand-ups so that the latest information is communicated.

The structure is very similar to daily stand-ups with the addition of two key questions: *'What output from your team in future sprints do*



*you see as possibly interfering with other teams' work?'* and *'Does your team see any interference problems coming from other teams' work?'*

Fig. 18 Scrum of Scrums

Unlike daily stand-ups, which are status updates, the Scrum of Scrums is about coordination and problem solving, so these questions of interference are resolved in the meeting (if there's time) or scheduled for resolution as soon as practicable.

## LeSS

Large Scale Scrum is a more radical, far less prescriptive approach to scaling than SAFe. It has gained some traction in medium to large companies and is supported by the Agile Alliance. Again, it's suitable for those building software and hardware/software products. A major focus is on centralized prioritization that's then distributed down the organization to those who are responsible for achieving local co-ordination and implementation.

The Product Manager supports the work of LeSS teams by helping

them understand the market and customers, as well as working with the team to prioritize the problems they will solve.

## DAD

Disciplined Agile Delivery builds on strategies and approaches from Agile Modelling, eXtreme Programming, Unified Process, Kanban, Lean Software development, 'Outside In Development' and others to create a hybrid that is suitable for medium to large organizations. It is a relatively new approach that is less prescriptive than SAFe. While this allows for more flexibility in dealing with the context of each business, it can also lead to challenges on how to implement what can appear as a set of disjointed approaches.

The role of the Product Manager in DAD is to focus on strategy, long-term product vision, and supporting outbound sales and marketing planning activities.

## Spotify

As Spotify grew from a small start-up working in a single room to employing over 4,000 people across multiple sites and time zones, they designed their own solutions to scaling so they could share best practice, align on architecture and agree on product priorities.

The terms they coined: Squads, Tribes, Trios, Guilds, Chapters, and Alliances have become widely known.

In brief, Squads are the equivalent of a Scrum team. Tribes are groups of Squads working on common functional areas. The group that helps keep alignment within a Tribe on the workflows and priorities is the Trio (the Tribe lead, a product lead, and a design lead).

A challenge when scaling teams is ensuring sharing of best practice and insights. That's the purpose of Guilds and Chapters. Chapters are for people working in the same area, e.g., Testers or Product Managers. Guilds are special interest groups where people can find out more and share knowledge, even if they have no responsibility in their role for the activities covered by the Guild.

Finally, sometimes there's a need for a larger group than a Tribe to work on a goal. In that case, an Alliance is formed (usually of 3 or more Tribes).



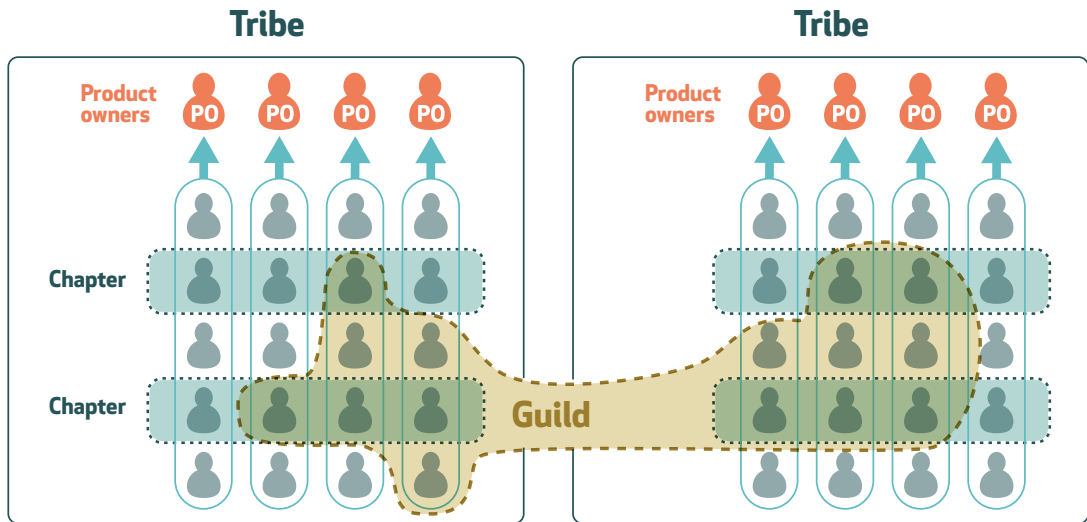


Fig. 19 Scaling at Spotify

The way of working was designed by and for Spotify – it's not right for everyone. For example, it depends on the teams being given clear objectives and boundaries, having the autonomy to create their own solutions, a technical architecture that supports this, and organizational acceptance that failing and learning fast is desirable. It also depends on having a culture that permits the flexibility it requires.

Unlike the other approaches listed here, there is no common training or certification. Even so, we know of companies that are attempting to learn lessons from Spotify and apply some of the same solutions in their business.

One other thing to bear in mind is that Spotify is learning all the time, so by the time you read this, squads, tribes, chapters, and guilds may be a thing of the past!

## Conclusion

Being more Agile is proven to bring benefits from improvements in customer satisfaction, delivery of business value, speed of delivery, quality improvements, staff engagement, and more.

It has its challenges, and even at a small scale, the ceremonies and processes don't always add value to every business and every type of development.

"A concern I have with many organizations that look to scale Agile is that they let "being Agile" become the goal. That's all backward. The goal should be to reach some outcome. Agile is just one means to help you get there effectively."

**Kent McDonald,**  
Product Manager  
& Founder,  
KBPMedia

# BEST PRACTICE

"Where Agile brings lots of benefits, it doesn't replace - but boosts - the importance of good product management. It is the outcome that matters to the business, not the sprint output."

**Edgar Wieringa,**  
Product Lead, Exact  
Online

Agile approaches can be scaled across development teams, but without changes to governance and other processes elsewhere in the organization, some of the potential benefits of agility will not be realized.

Scaling across an entire organization, i.e., Agile transformation has significant additional challenges but achieving the benefits of more agility across the whole company is a prize that may well justify the effort.

While scaling across the whole business isn't commonplace, neither is it at the 'bleeding edge' of organizational design. There are proven approaches, experienced people as well as training and certifications that can help build the knowledge to succeed.

However, whatever Agile scaling approach is used, the need for Product Managers to define product strategy and roadmaps, understand markets and customers and take a lead for their product is as important as ever.

# Tools

Software to help you manage requirements and projects

In the early days of Agile, many companies had dedicated 'War Rooms' where the team tracked work using notes stuck to the wall. It was great for team spirit to have everyone able to physically move their post-it notes as tasks progressed from *To do* to *Doing* and finally to mark them as *Done*. However, this approach made it tough to communicate progress to a wider group of stakeholders, difficult to track interdependencies, and almost impossible when teams worked in different locations and time zones.



To overcome these problems, most teams now use software tools. These enable the Product Owner to keep track of the source of requirements, manage their backlog, and track commitments. The Agile team can make estimations, understand interdependencies and follow progress. Senior stakeholders can get the big picture.

Most tools offer functionality that's useful for Product Managers and Owners. There are tools specifically for generating ideas, others for roadmapping, and others for visualizing user interfaces. Some can take you all the way through from creating ideas to generating a roadmap.

Our 2022 Survey of the Product Management Profession has a section on the most popular tools used in the industry and the Top 10 Recommended Tools, based on data from over 1,300 product people.



# A guide to Zen Agile

or "Sh\*t happens, get over it"

*This is an abridged version of a blog written by Aidan Dunphy. The full version can be found [here](#).*

"Agile" is one of the most discussed topics in tech. You might say that we're in a 'post-Agile' era, with it being 'de rigueur' for software development.

However, just as much is written about the tendency of senior management to think that Agile is a magic trick, enabling teams to do more with less (and faster) rather than something they themselves need to adopt.

Is agility a higher state attainable only by elite outfits like Spotify, or is it something that any company can achieve?

## So, what is this Agile thing anyway?

It's important to remember that Scrum and Agile are different things – it is perfectly possible to use Scrum without being Agile, and vice versa. So, when Drift CEO David Cancel shouts, "I hate Agile," perhaps what he means is: 'I hate when people do a Scrum thing and think that means they're Agile!' Or put it another way. Agile as an industry of training, consultants, ceremonies, and artifacts doesn't necessarily translate into

businesses becoming agile, nimble, and flexible.

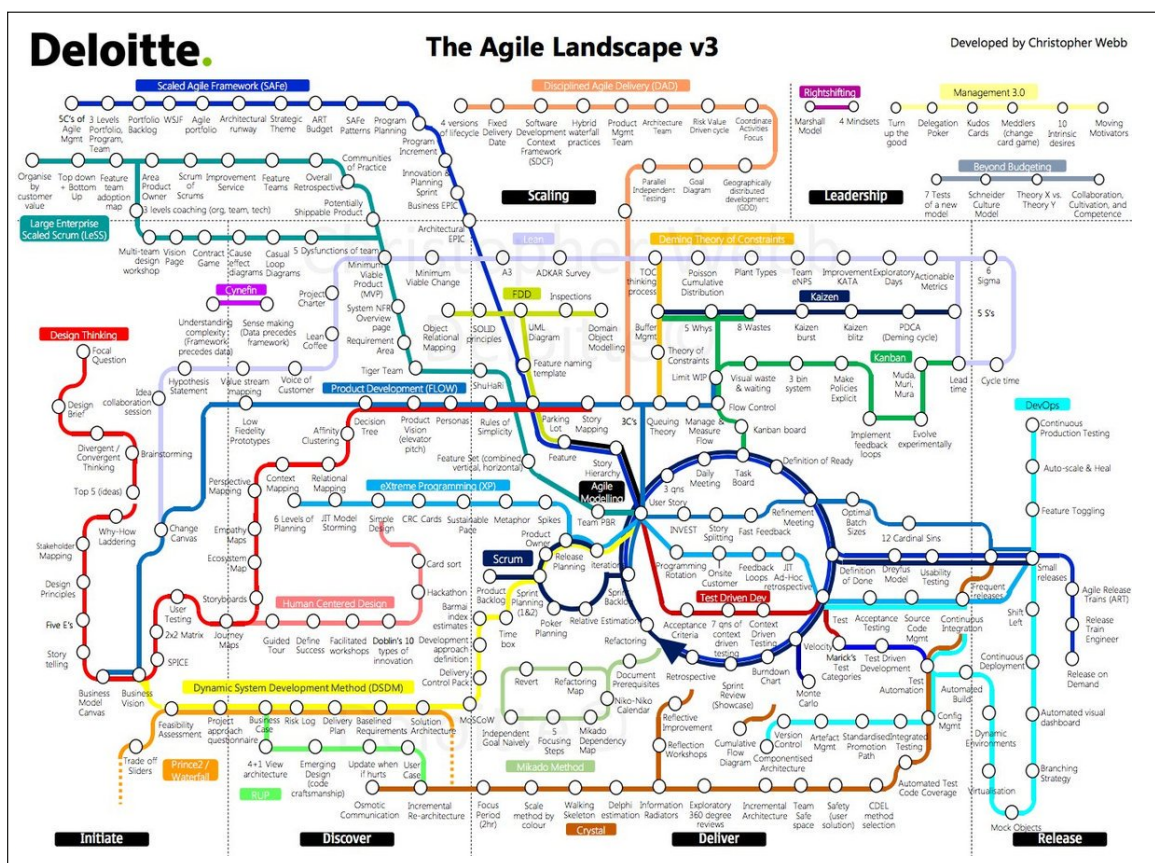
In 2016, a Deloitte analyst inadvertently portrayed this maelstrom of jargon in his infamous 'tube map' diagram. This diagram showed (some of) the various tools and practices advocated in the Agile world.

Ugh – easy to see why someone might think Agile has lost its way.

Agile is something you should be, not something you should do, and no tool will make you Agile.







## Let me give you some personal examples...

A former CEO I worked for couldn't get his head around why I'd sometimes reposition halfway through a meeting. His preference was to relentlessly persevere with "the plan," an always-out-of-reach goal to "arrive in style." Needless to say, he was a former project manager from whom the word "Agile" literally elicited a wince.

Living with such constraints is an ordeal for Agile thinkers.

As a university student, I was once rushing to catch a train home to Durham at the end of a busy week. I ran into the station and onto what I thought was the 17:25 waiting at platform 2. Imagine my dismay when we departed in the wrong direction, and the guard then confirmed that the first stop was Edinburgh (in 2 hours). My half-hour commute had just turned into a 4 hour round trip to Scotland.

Fig. 20 The Agile Landscape. Source: Chris Webb, Deloitte. LAST Conference 2016

# WATERFALL VS AGILE

Most of it was spent sitting in an empty carriage, and with the darkness outside, I couldn't even admire the scenery. Not planning for this impromptu trip, I had nothing to read other than the train company's brochure, urging me to take a day trip to historic Durham. It's not unlike working in a project-driven company and reading blogs about A/Agile.

On another occasion (stay with me), when working as a consultant, I rushed out of the office, jumped in the car, and was bombing southward from Newcastle on the A1. As I passed the Scotch Corner services, a sudden bolt of awareness struck cold fear in my spine: Where was I going? Stopping, I got my laptop out to check my diary (no smartphones yet) and confirmed that I should have been heading North to Glenrothes. Fortunately, the unintended detour only added 90 minutes to my journey.



Did you spot the analogy?

The train is Waterfall and the car, Agile. In both cases, I took the wrong initial course. But, on the train journey, I had to continue until the bitter end.

I did ask the guard if he would stop the train to let me off, but he just laughed at me with the incredulity of a PRINCE2 program manager when you ask them to pivot.

For me, being anything other than Agile is effectively to pretend that (a) you can predict the future and (b) sticking to a bad plan is better than changing it.

So why does Waterfall still exist, given its dire reputation in software product development?

## Letting go of the future is counter-intuitive

We, humans, are instinctively able to discern patterns in things. We

# SHOW ME THE FEATURES

use this to foretell the future by extrapolating from our past experiences. Side effects of this include an emotional need to know what's going to happen and a predisposition to confirmation bias. Doing a familiar task makes you feel like you're on safe ground, and you can go a long way along the road to damnation before reality intervenes.

So, when planning a Waterfall project, everyone involved conspires to delude themselves that they're going to release 'on time' and 'on budget'. Execs push a date at which the release must happen, and delivery teams make promises based on estimates. Devs complain about this, pointing out that it's impossible to accurately forecast software development, so someone adds 'contingency' to the made-up numbers. Everyone starts out feeling comfortable and positive because there is a plan.

Before long, however, stakeholders start to wonder what's happening. As there is, as yet, no software to be seen, their nerves start to jangle about the pace of progress. They enquire into how things are going, only to be told that "There's nothing to see yet; we're still architecting the platform."

The lack of visible progress on a Waterfall project can lead to some nervy stakeholders. The anxiety builds to the point where they demand to know whether the deadline is going to be met and that something tangible is produced. The delivery team grumbles about 'scope creep,' 'changing requirements,' and having to make 'shiny stuff'. This negativity is driven by a growing feeling that control is slipping from their grasp.

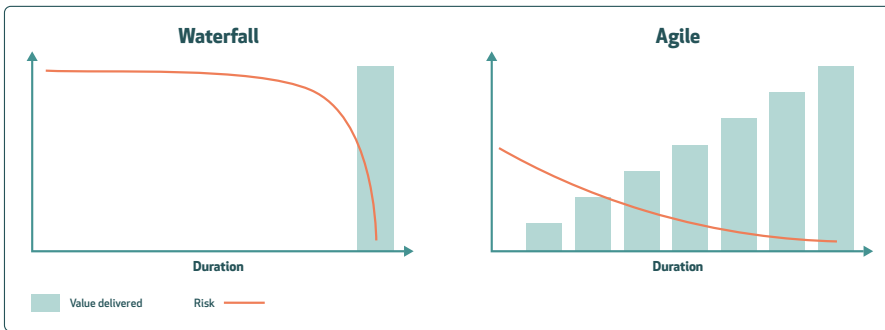
At the same time, they can tell the suspicion is growing from people outside the team who 'just don't understand how hard this is'. The deadline is extended, tempers fray, additional pressure is heaped on the team, and people quit. We've all seen it before. When running a project this way, you should start out by writing the post-mortem!



Source: Edited still from Jerry McGuire movie Sony Pictures

In truth, it's not feasible to accurately predict how a software project will pan out, as the control was never really there in the first place. This is the main reason for the 'inspect and adapt' concept in Agile.

It encourages the delivery team to review their progress early and often, thus reducing the risk of doing the wrong thing for a long time.



You may have seen the following model for comparing risk between Agile and Waterfall methods.

The idea is that in a Waterfall project, the risk remains high until

delivery. Whereas, in Agile, the risk is reduced from an early stage by iterative inspection and adaptation.

However, this analysis of Waterfall is rather optimistic. In practice, it often creates a graceful, slow-motion failure founded on stated prejudices (AKA "requirements").

A bulletproof change control procedure will cover the Project Manager and even make them look good. The aim is not to create value; it's to ensure that a plan is followed. Boxes ticked, meetings minuted, excuses made. All too often, the outcome is an unrecoverable failure, and having spent so long building the wrong thing, there's no budget left to rectify it. First-mover advantage has been lost.

### The deadline delusion

It's Agile blasphemy, but also an inconvenient truth, that deadlines are inevitable in business.

Building a product costs money, and there is generally a finite amount of it available. Therefore, stakeholders are understandably keen to spend as little time as possible spending it. Unfortunately, in software projects, deadlines are nearly always missed. This can be because:

Fig. 21 Value vs. risk comparison

- Customers always want stuff yesterday, and your competitor is threatening to give it to them, leading to pressure to say “Yes, soon.”
- Technical people often overestimate their abilities and neglect to account for mistakes, lost time, and unforeseen circumstances.
- Once the team is making something, they feel good and fall into the trap of thinking there is plenty of time left. Only when things are obviously off track does anyone intervene, and by then, it’s too late.

In fact, if you meet a deadline, it’s probably because the stakeholders were prepared to overpay to guarantee it.

## Transcending the tyranny

How to escape this?

Permit me one more confessional anecdote: As a heavy smoker in my twenties, I resisted giving up for many years. I was unable to countenance a future as a clean-living killjoy and felt that my life’s course was already set. When I finally decided to quit, it was genuinely terrifying for me because it felt like dying. Based on the smoker/rebel template, my personality was ending, and I didn’t know whether there was life after the death of this self-image. It was a traumatic experience, but I survived (obv), and this major version upgrade taught me some life lessons.



## Lesson 1: The only thing that really matters is what happens next

Breaking my self-fulfilling prophecy that I’d always be a smoker was hard. I’d constructed a mental life script for myself, with a sense of inevitability based on my sense of self.

Unfortunately, human intuition, which generally serves us well, also suffers from systemic faults such as the sunk cost fallacy. We tend to persevere with a fruitless course of action because of the cost already incurred. We think, “I’ve had the bad luck; some good is due soon!” Nope, any money you’ve wasted on a product is now in the past, and the only factors governing your future success are happening right now.

Like Airbnb, great products came about as a result of many sprints, with many mistakes and brave decisions made along the way. A stretched analogy, perhaps, but life and business are anything but deterministic.



# LESSONS

## Wise Words

"Do not condemn the judgement of another because it differs from your own. You may both be wrong."

Dandamis,  
philosopher

### Lesson 2: If you decide to keep going for something, you'll probably get there, eventually

One day, around a year after quitting smoking, I realized that I was no longer thinking about it all the time and had no inclination to go back. This only happened because I had decided on 17th March 2001 at 13:35 GMT to become a non-smoker. A definite goal, but with an uncertain time scale.

Similarly, my car trip had a destination (despite the detour to Scotch Corner). If there had been no 'north star' goal, I might have found myself endlessly negotiating the one-way system in Middlesbrough – keeping very busy but not actually going anywhere. To paraphrase Lewis Carroll, "If you don't know where you're going, any road will take you there."

Agile is not random. Rather, it's mindful pragmatism that keeps you moving in the right direction. You continually apply course corrections, but always with a destination in mind.

Fans of classical project management will argue that this can be handled via change control. However, such frameworks are predicated on the assumption that change is an exception.

Conversely, you can think of Agile as continuous change control. Although, we shouldn't talk about "change" and instead recognize and indeed embrace uncertainty.

Throughout my career, I've heard delivery teams complain about changing requirements and "scope creep." In most cases, the client hadn't really changed their requirements. It was just that the implications of their vague ambitions had become progressively clearer on closer inspection and as time passed.

### Lesson 3: Fear causes all the trouble

Channeling that great business philosophy guru, Yoda: "Fear leads to negativity; negativity leads to inertia; inertia leads to poor quarterly results."

The reality is that it doesn't matter how much you plan; things aren't really under your control. There's no point being afraid. What's the worst that can happen? Well, you could lose your job? Faced with this worst-case scenario, it's worth asking yourself, "If I'm miserable here because of constrained thinking, lack of trust, and systemic fear, shouldn't I go somewhere else?"

However, it is disruptive to have to keep switching jobs. So, how can you overcome the deadline delusions and control freakery?

The trick is to understand that stakeholders are also prone to fear. They usually don't grasp Agile and expect to see a linear progression, such as you might see with a house taking shape on a building site. However, making a new product is far less predictable than making a building, and it's important to understand how stakeholder confidence changes over time:

Fig. 22 shows how stakeholder confidence may develop over the lifecycle of a project.

In the Waterfall case,

confidence will be either restored or shattered completely at release time, depending on whether the product meets the expectations that have built up over the project.

In the Agile case, early confidence can be dented as incomplete stages of the product are produced by the delivery team. But, so long as the team keeps taking feedback, confidence builds as subsequent iterations move visibly closer to a market-ready product.

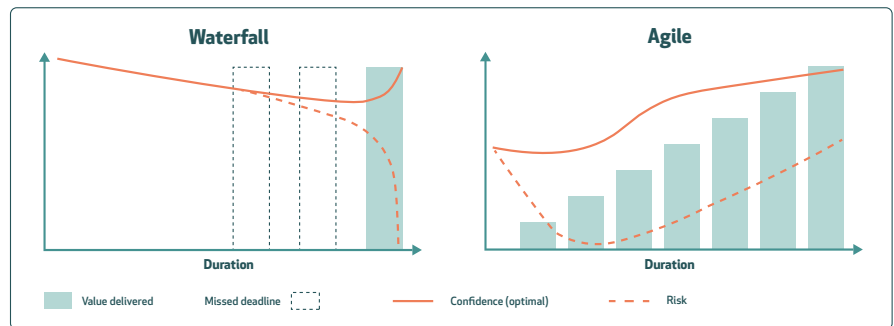


Fig. 22 Stakeholder confidence over the life of a project

### A plan is not a goal – it's a tool, and an adjustable one at that

If you're building a business and need some software made, you will probably have some idea of what you want your product to be. You will also need assurances from your product team that they can build it in a reasonable timescale without busting your budget.

So, how to fix the three corners of the scope/resource/time triangle?

Traditionally, this would be achieved by spending a lot of time analyzing the requirements in detail, planning exactly what's going to be built, estimating it, then adding contingency and a change management process. If a high degree of confidence in the time and cost is required,

the price is often accepting a much longer timescale and a higher risk that the eventual solution will fail to match requirements that emerge or change during the project.

Agile is often seen as a silver bullet. Don't plan beyond one sprint! Pivot every now and again! Prioritize everything brutally! Que sera sera! Zealots will shout down any talk of roadmaps, planning, or strategy.

Of course, I'm being a bit silly, and nobody would take this seriously as a business approach (you would think). Businesses happen because someone had a vision and pursued it with vigor and persistence in the face of adversity. It is the founder or business leader who takes a punt on and is accountable for an idea. Agile is great for handling a change of direction in the detail or even means of execution. But, if the big idea fails, the business is gone.

### **Relax, nothing is under control**

To sum up with a Zen-like platitude...

Beware the duality trap, in which one believes they can plan the future of a product.

Learn to accept the reality of the current sprint; appreciate its innate business value.

And, be comforted by your career prospects, which will transcend the noise of ephemeral shouty bosses if you keep adding value.

### ***Author's bio***

*Aidan is a Product Focus Senior Consultant.*

*He has held Head of Product, Product Director, and Chief Product Officer roles in technology vendors and digital agencies within a broad range of industries.*

*Aidan has in-depth experience of the full software lifecycle, including start-ups, scale-ups, and mature product portfolio management.*





# PM/Agile fit

The broad role of product management

**Agile methodologies like Scrum are used widely to develop software products.** They describe key roles and responsibilities for the product development team with the aim, as discussed elsewhere in this journal, of delivering products that are valued by customers.

What is not included by most Agile methodologies are the other activities needed to ensure products are successful, for example, optimizing in-life, pricing, and marketing. Without these activities, a great product, while it might be valued by customers, won't maximize its commercial return for a business and may fail altogether.

So, what's needed by businesses are the activities covered by Agile development methodologies as well as other product management activities.

## The impact of Agile on ways of working

As discussed elsewhere, Agile enables the rapid validation of product ideas. It accepts that we can't know the right answer upfront and that change is inevitable – so we are better off building iteratively and incrementally.

If we're managing a product, this impacts the cadence that we work at and our workload. We need to work closely with Development on a continuous basis, but we get fast regular deliveries that we can try with customers, and there is less complaint when we make a change in direction.

It's also about empowerment. Trusting product teams to deliver products that customers want and giving them the tools and authority to get on and do it.

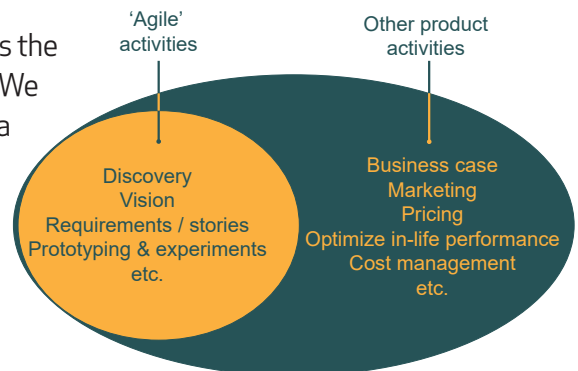


Fig. 23 Successful products

# PRODUCT ACTIVITY FRAMEWORK

## What does product management include?

In the product world, we find that it's almost impossible to guess what someone does from their job title. It varies from company to company and sometimes even within the same company.

So, it's important to distinguish between job titles and the activities they actually do.

In our Product Activities Framework, you can see all the activities that need to be done in any company with products.

The framework is divided into three key areas – **strategic product activities**, which are about working out the right product for the business. **Inbound activities** which are about helping the business to

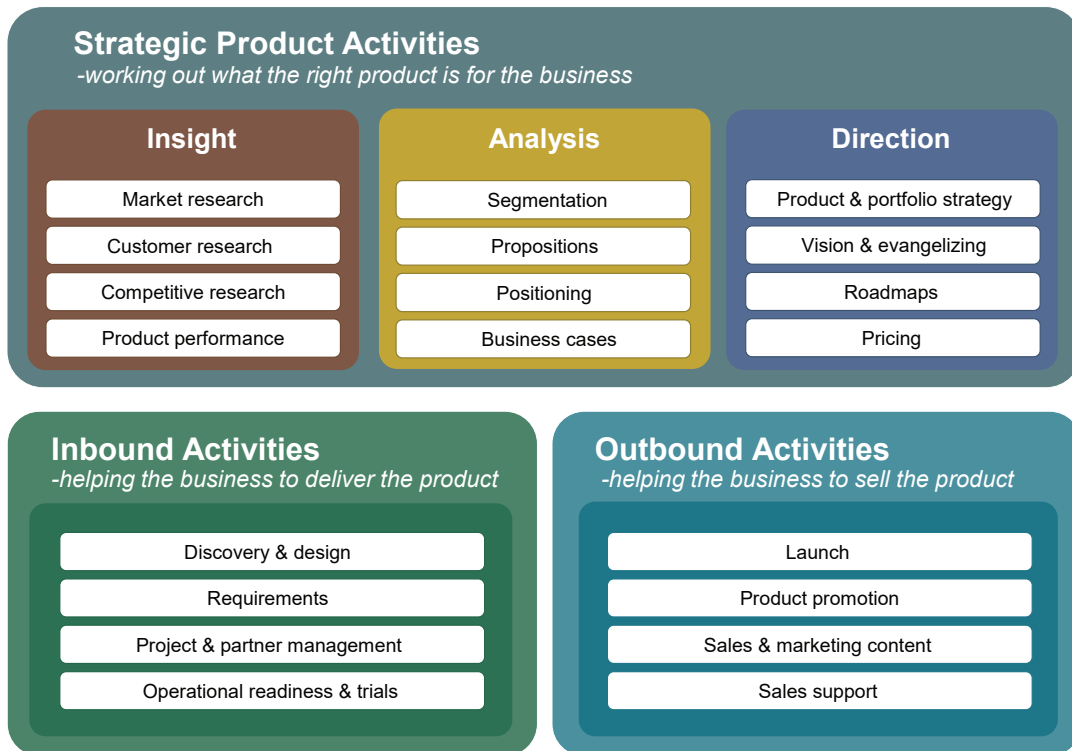


Fig. 24 The Product Activities Framework

develop and support the product. Finally, there are **outbound activities** which are helping the business to market and sell the product. Product Managers take on many of these product activities for their product.

# ORGANIZATIONAL STRUCTURE

## Product Owner or Product Manager?

In some product businesses that use Agile, there are Product Owners and no Product Managers. But, that doesn't mean these product activities aren't being done. Product Owners, as defined by Scrum, are responsible for maximizing the value of the product resulting from the work of the Development team, which typically maps to the Inbound Activities in the Product Activities Framework, coupled with work on vision, customer insight, and planning. When there are no Product Managers, we often see that the Product Owner responsibilities expand to take on some of the other activities shown in the framework.

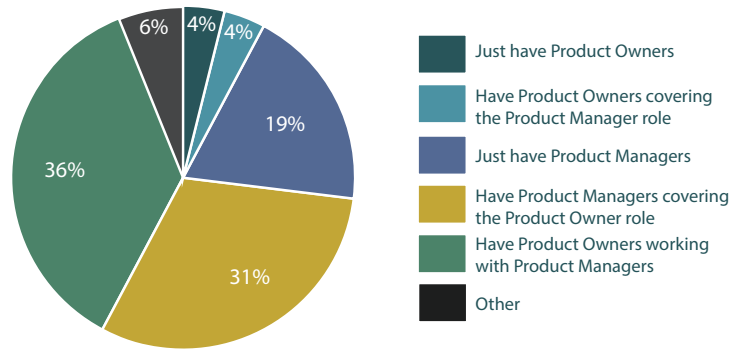


Fig. 25 Who decides?

More often, we see the product activities split between a Product Manager and Product Owner role – with typically the Product Manager giving high-level product direction to the Product Owner. You can see who makes the decision to build from our 2022 annual industry survey (Fig. 25).

## What is the impact on organizational structure?

There are many different organizational structures for companies.

Most 'traditional' companies have a matrix organizational structure with separate functions and departments (Development, Testing/QA, Finance, Marketing, Product Management, etc.) and then temporary cross-functional teams with a mix of different roles that come together to work on products and projects.

You can think of these as vertical and horizontal dimensions. These companies prioritize the functions, which are shown as the vertical dimension in Fig. 26.

"The Scrum title Product Owner has always been problematic. Product Owners don't own the product. If there's anything they do own, it's the backlog and working with the team to prioritize work being done. And that's how the role was originally defined. Over the years, Scrum advocates have expanded that to include things like budget, strategy, etc. It's caused more problems than it's solved, in my opinion."

**Saeed Khan, Industry expert and founder of Transformation Labs**

It makes sense to keep the experts from one function together so they can learn from each other, build expertise and the company can manage scarce and valuable resources. For example, now and again, a Product Manager needs support from Finance to develop a business case, but the product doesn't need this expertise available full time, all the time.

Organizational structure prioritized by function

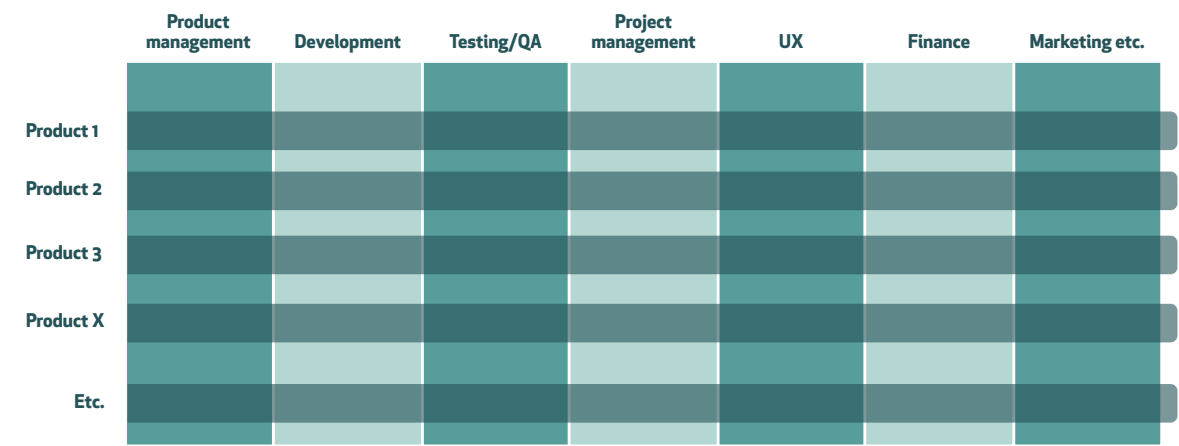


Fig. 26 Structured by function

In this structure, a Product Manager typically leads a virtual team with representatives from the different functions. For example, when launching a product, they may be working with colleagues from the Development, Sales, Marketing, and Support functions. These colleagues don't report to them directly, but like a sports team, the Product Manager is the captain.

Resources, virtual teams, and silos

In this more traditional matrix-style organization, people are assigned from a function for the duration of a project into a temporary team. It is founded on the myth that people are fungible (exchangeable) and reinforced by using the word "resource" for people. Essentially the idea is that you can put a different technical architect onto a product, and they will be as productive as the last one because they have the skills. What this misses is that someone new doesn't have the tacit knowledge (that only comes with experience) about that product and its domain.

Another problem with this organizational design is that although the horizontal dimension is deemphasized, it represents the value of delivering teams that are key to the company's success. Various techniques are used to support this horizontal dimension like cross-functional processes (e.g., for product development) and having objectives, reporting, and incentives for the product or project teams.



And, of course, having separate functions can lead to a silo mentality – especially if each function has its own objectives. If you've got a Product Manager in one department and a Product Owner in another, it's easy to see how they might start pulling in different directions.

## Organizational structure prioritized by product

So, another approach is to make product teams the dominant vertical dimension.

Although not radically new, permanent cross-

functional teams are a key tenet of Agile, and many digital Agile organizations (like Spotify) are set up like this. See the article on Scaling Agile on page 30.

With this approach, the product team (or squad) is designed to feel like a mini start-up. They have all the skills and tools needed to design, develop, test and release a product. They can really become experts in their product area and feel strong ownership of what they develop. They all sit together in the same team, pull in the same direction, and because of that, their performance is more predictable.

## Organizational structure prioritized by product

	Product team 1	Product team 2	Product team 3	Etc.
Product owner				
Developer				
User experience				
Tester				
Etc.				

Fig. 27 Structured by product

# DIGITAL TRANSFORMATION



This resonates well in the digital age. Google and many other leading tech companies use this cross-functional or team-based organizational structure. It's not about top-down orders but empowering teams to deliver. It helps maintain a small-company feel in a big company and promotes the notion that all employees play an equally important part in the company's success. Culturally, it places more importance on intelligence and ideas than on titles ... *'democracy and meritocracy rather than command and control.'* All this plays well when you're employing highly paid knowledge workers who can easily move to a competitor, should they want to. And more importantly, it creates great software products!

So what does it all mean for product management?

"The team has grown globally and is a mix of contractors and internal people. To ensure confidentiality, we have adopted an approach where our stand-ups may be divided into two parts, starting with the contractors contributions/questions and then moving on to the internal team. The sprint review and retrospective are performed with a full team unless sensitive information is shared. Also, sprint planning is done with a full team."

**Geir Olsen, Product Owner, Rogaland Maritime Service**

## Most organizational structures are a mix

Most, if not all, organizations with technology products have a mix of product teams and separate functional silos. Development may happen in the product teams, but product management activities get done across the business by a group of people with various job titles, including Product Manager.

And there are always real-world constraints – resource and budget limitations. So Product Owners take on multiple scrum teams, scarce UX resource is spread across development and subject matter experts, e.g., Agile coaches, are spread across the organization.

In our experience, organizational structures grow organically. Organizational design and Agile consultants come and go ... empires wax and wane. That's not necessarily bad as organizational structure (like everything else) need to adapt and change as a business evolves.

## Is Agile the same as 'digital transformation'?

Digital transformation is the use of digital technology to help deliver more customer and business value. This implies using software and creating digital products. As Agile development approaches are almost universally used for software, the two go hand-in-hand.

But digital transformation can also apply to companies with physical products. It might be the digitization of their supply chain tracking or order in-take, just as much as digital products' delivery. For example,

Domino's Pizza developing a digital ordering platform that gives customers visibility from their order through to delivery.

Product management should be at the heart of this as we are a key catalyst for this sort of change. With our focus on customer and business value, this is what we should be championing.

## Conclusion

The reality for most product people is that they don't get to choose what their organization looks like. Development introduces Agile development approaches; digital transformations are spearheaded by the CEO and the organization changes. If you're a senior product person, you may be able to influence things, and even if you're more junior, you can still influence best practice in your area.

But, we believe, whatever your organizational structure or job title, you need to make sure you do the most professional job you can. And, world class product management is needed in organizations whether they use Agile or not.

"I provide consultancy services to businesses (largely business to business), advising them on how they can adopt a more customer-centric approach centered around the management of the product lifecycle, from idea through to retirement. I frequently come across the cultural clash between traditional governance models and the desire for the organization to be "Agile"; in fact, it is often commented that "Agile" and "governance" can't co-exist, leading to organizations being unable to embrace Agile as leaders/managers cannot countenance "losing control."

This is a problem that goes beyond whether or not to use an Agile methodology (Scrum/Kanban, etc.) in a product development environment and into the broader business approach and organizational culture. I find that Agile software development methodologies can work perfectly well in a more "traditional" business environment, provided that the organization accepts the reality of uncertainty and accepts Agile approaches as the practical solution to this challenge.

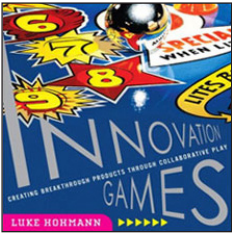
If the Agile approach is trusted and supported both by the people who make the decisions and the processes that lead to those decisions, then it can be incredibly effective."

**Phil Duffy, Director P&P Consulting**

# The Review

Reviews and feedback for Product Managers

## Book Review



**Innovation Games**  
by Luke Hohmann  
(Addison Wesley,  
2006)

## Wise Words

"You'll know a game is going really well when your customers don't want to stop.... it's this deep level of engagement that gets past any barriers to communications and produces the most honest and useful feedback."

**Luke Hohmann**

**Whatever development process you are using, discovering customer requirements is a key part of the product management role.** Most of us take the standard approach of customer meetings, market research, and talking to sales and support. However, getting real customer insights is usually a challenge.

Innovation games are a fun and effective way of working with customers to understand their needs, wants, and desires. They are an alternative to simplistic questioning, which tends to influence the discussion or costly market research where an agency interprets the responses for you. Customers often have a hard time articulating their problems or envisioning solutions, and these games draw out insights and nuances that can be extremely valuable.

The book's tagline summarizes its goal of creating breakthrough products through collaborative play. It contains 12 different games with a description of each, why it works, how to prepare for it, and how to process the results.

One of the games is called Product Box. You ask customers to imagine they are selling your product at a trade show. You give them some blank cardboard boxes, pens, and colored paper and ask them to design the box packaging. The box might show marketing slogans, features, pricing, or pictures. When they're finished, they have to present the box back to the room and persuade the other players of the features and benefits. This gives customers a way to tap into their latent preferences and requirements and express them as they sell your product back to you.

Of course, one of the potential drawbacks of these games is getting hold of the customers to participate. And the next thing to consider is whether you feel confident to run the innovation games yourself or want to bring in some outside consultants to help. However, once you've addressed these issues, the games are a low-risk and engaging way of getting high-quality insights into your customers' requirements.



# Agile - Beyond the BS

*Agile is simply analogous to a biological or electronic feedback mechanism, so...*

1. Find out where you are
2. Take a small step (action) towards your goal
3. Measure the result (response)
4. Adjust your understanding (behavior) based on what you learned
5. Repeat



*But, for this to work, we need to know our clearly defined goal. When faced with two or more alternatives that deliver roughly the same value towards the goal, take the path that makes future (code) change (and development) easier.*

*That's it, nothing more... nothing else is needed other than the necessary common-sense execution skills (design, planning, programming, and communication).*

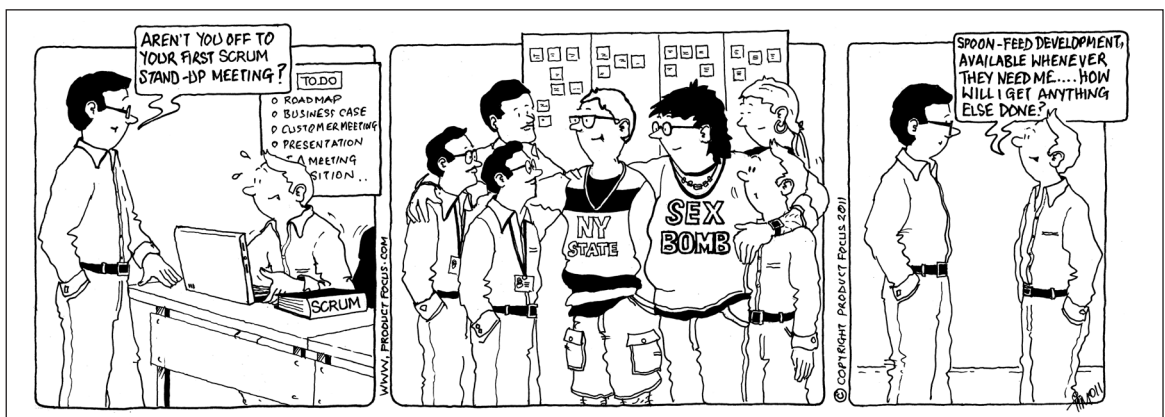
*There is no need to make an industry out of Agile and SCRUM and complicate it with rules and diagrams, with certification, training programs, and other controlling and money-making 'needs' of the software 'industry'. Agile is not a state but a process.*

**Paul Vallance, Menantol**

The full article can be read here on [menantol.co.uk](http://menantol.co.uk)

## Annual Survey

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



# The Insight

## What's the point of PM in Agile?

**We know that many companies don't have anyone with the title of Product Manager.** They might have Product Owners, Proposition Managers, User eXperience leads, but no Product Manager. Have they got it wrong?



The fact is that if you've got products in your business, then product management activities will (or should) be getting done somewhere – even if you don't have Product Managers.

One of the most important of these is **product leadership**. Various Agile approaches propose self-organizing teams, collaborative decision-making, and shared ownership. Sounds good... until it goes wrong.

What happens when people are at loggerheads? What happens when resources or time are tight or when there's uncertainty over which direction to take the product?

We've seen examples in Agile companies where things just grind to a halt, decision-making gets handled by constant escalations, or there are compromise decisions that attempt to satisfy everyone but are not best for the product or the business.

Even in the area of product discovery, where constraints should be minimized, some boundaries are needed to ensure resources are spent wisely and achieve the best outcome. It might be as limited as someone defining the market segment, the customer roles that should be involved, the geographies where you'll launch, or the product vision with which the discovery work should broadly fit.

Following that lead, other roles can align their work and bring their specific technical, user experience, or marketing skills to bear.

What role should do this?

While we believe that the best job title for this role is Product Manager, frankly, the title doesn't really matter.

If you've got products, someone needs to be actively managing them. Someone needs to take on product leadership, and the fact that someone is doing this needs to be clear and accepted by everyone else.

### Wise Words

"If you hype something and it succeeds, you're a genius... if you hype it and it fails, then it was just hype."

**Neil Bogart,**  
American record  
executive

# Training and Support for product management leaders



Learn how to manage a product management function and team

Start up or improve product management with a Product Focus Review

Explain the value of PM with an Executive Briefing

product

focus


[www.productfocus.com](http://www.productfocus.com)

product

focus

# **Learn best practice and improve performance with the European leaders**

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

 **+44 (0) 207 099 5567**

 **info@productfocus.com**

 **www.productfocus.com**