ROMAN PICHLER

STRATEGIZE

Product Strategy and Product Roadmap Practices for the Digital Age

Second Edition

STRATEGIZE

Product Strategy and Product Roadmap Practices for the Digital Age

SECOND EDITION

ROMAN PICHLER

Strategize: Product Strategy and Product Roadmap Practices for the Digital Age, 2nd Edition

Roman Pichler

Published by Pichler Consulting.

ISBN: 978-0-9934992-5-8 (Kindle)
ISBN: 978-0-9934992-6-5 (ePUB)

Design: Ole H. Størksen
Layout: Booknook
Cover photo: Shutterstock

Digital book(s) produced by Booknook.biz

TABLE OF CONTENTS

PREFACE

Progress is impossible without change, and those who cannot change their minds cannot change anything.
George Bernard Shaw

Developing a successful product is not down to luck, a stroke of genius, or just trying hard enough. While these factors are undoubtedly helpful, product success starts with making the right strategic decisions. A challenge for many product managers, product owners, and other product people is that we are often so preoccupied with the tactics—be it writing new user stories or dealing with an urgent support request—that we sometimes no longer see the wood for the trees. In the worst case, we take our product down the wrong path and end up in the wrong forest; we've perfectly executed the wrong strategy and are left with a product that underperforms or even bombs.

Why You Should Read this Book

This book will help you play a proactive game, make the right strategic decisions, use them to guide execution, and help you maximise the chances of creating a successful product. It explains how to create an inspiring product vision, a winning product strategy, and an actionable product roadmap for digital products that are developed using agile practices. The book offers a wide range of carefully selected, proven techniques and tools that complement and support each other. If you work as a product manager, Scrum product owner, product portfolio manager, head of product, or product coach, then this book is for you.

A Brief Guide to this Book

Book Structure and Contents

This book begins with an introduction followed by four chapters on product strategy and three chapters on product roadmaps. While you might be tempted to go straight to the sections you are most interested in, I recommend reading the introduction first. It presents central ideas, and it truly forms the foundation for the remainder of the book. Reading it will provide you with context to correctly understand the individual chapters and sections. While I have tried to write the book so that the chapters are loosely coupled, there are dependencies between them. Most notably, the roadmap chapters build on the strategy ones, as a realistic and actionable product roadmap should be based on a validated product strategy.

Please note that this book is neither an introduction to product management nor to agile practices—there are other great books that equip you with this knowledge. Instead, I assume that you are already familiar with central product management concepts and an agile framework like Scrum.

Sources, Examples and Terms Used

This book draws on a range of approaches including Lean Startup, Customer Development, Design Thinking, the Blue Ocean Theory, Business Modelling, Jobs to be Done, User-Centred Design, Lean Management, Scrum, and Kanban—in addition to my own methods and templates. I have taken great care, though, to combine the different concepts, techniques, and tools in such a way that they fit together and complement each other.

Most of the examples in this book are taken from the consumer space. The reason for this is simple: I have tried to use products that I hope you have heard of. But the practices covered are applicable to virtually any digital product, including B2B offerings. You can also apply most of my advice to non-digital products. You might have to adjust some of the practices, though, and ignore the software-specific recommendations.

Finally, I refer to individuals who manage a product as *product person* and *product people* rather than *product manager* or *Scrum product owner* to avoid any negative connotations some readers might associate with these terms.

Changes in the Second Edition

This edition of *Strategize* updates and enhances the first one, which was published in 2016. It offers new concepts, adds new materials, shares more tips, provides new examples and illustrations, expands and details existing content, and simplifies the book structure. I believe that this has made the book even better and its advice easier to apply.

INTRODUCTION

If you don't know where you are going, you'll end up someplace else.
Yogi Berra

My first product management experience wasn't exactly what you call a success: I was working as part of a team who was called in to help with a new product development effort, and I ended up working with the product manager leading the product. While I learnt a lot in the process, the resulting product sadly failed. But this taught me an important lesson: There is no point in worrying about the product details and writing use cases and user stories if a sound product strategy is missing and an actionable product roadmap is not available. Strategy and roadmap are truly fundamental, as I explain in this chapter.

A Product Strategy and Roadmap Model

An effective product strategy and roadmap are crucial to successfully create, enhance, and manage a product. But what exactly is a product strategy and what is a product roadmap? How do the two plans relate? And what's their relationship to the product vision and the product backlog? To answer these questions, I have developed the model shown in Figure 1.

Let's take a closer look at the model and explore its elements and connections in the next few paragraphs. Please note that I'll discuss the detailed practices to work with the artefacts in Figure 1 in the remainder of this book. The following sections therefore provide an overview.

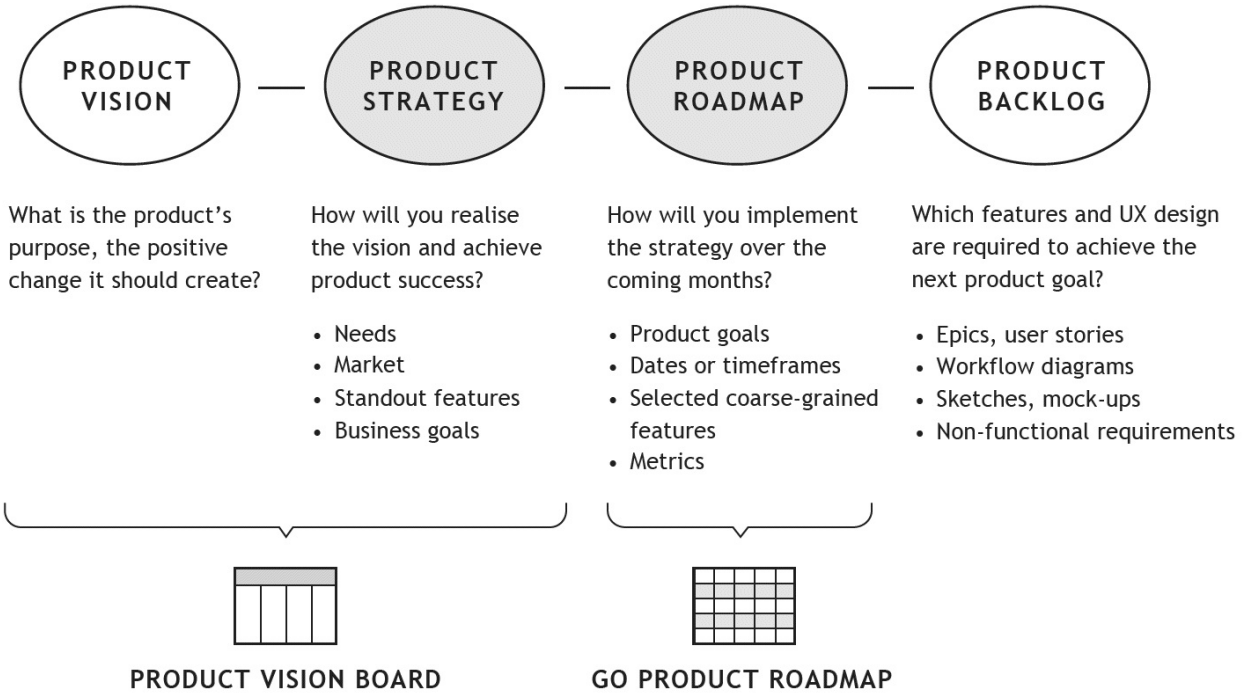


Figure 1: A Product Strategy and Roadmap Model

The Elements of the Model

At the heart of the model in Figure 1 are four artefacts: the product vision, the product strategy, the product roadmap, and the product backlog. The *product vision* describes the product's purpose, the ultimate reason for creating it, and the positive change it should bring about. You can think of the vision as the product's true north that guides and aligns everyone involved in achieving product success. This includes the stakeholders, the management sponsor, and development teams.[1]

The *product strategy* communicates the approach chosen to realise the vision and to make the product successful. Coming up with a strategy requires you to make four important choices: selecting the needs the product should address; determining the market or market segment—the users and customers who should benefit from the product; choosing standout features that set it apart from competing offerings; and setting realistic business goals that describe the benefits the product will create for the business. Making these choices requires you to say no to ideas and suggestions. While this can be tough, it is a necessary part of strategic decision-making. A product that tries to please everyone risks not doing a good job for anybody. What's more, a new or significantly changed strategy must be *validated* to maximise the chances of achieving product success. This is best done by systematically addressing its biggest assumptions and risks.

With a validated product strategy in place, you are in a great position to build an actionable product roadmap. The roadmap describes how the product strategy will be implemented in the next six to twelve months; it communicates the specific benefits the product will achieve; and it aligns and guides the stakeholders and development teams. An effective product roadmap is built on *product goals*, which describe the outcomes the

product should create, for example, acquiring new users and increasing engagement.[2] The roadmap may also contain additional elements like *dates* or *time frames*, selected coarse-grained *features*, and *metrics*. A date or time frame states when a goal should be met, the features sketch the output required to achieve a goal, and the metrics help you understand if a goal has been accomplished.

A product roadmap that is built on product goals provides a great basis for deriving a product backlog and making the right tactical product decisions. You can simply copy the next product goal into the backlog together with its features. Then add further items that are required to meet the goal, such as epics, user stories, workflow diagrams, sketches, mock-ups, and non-functional requirements (NFRs).

In addition to the four elements described above, my model offers two templates that I have created, the *product vision board* and the *GO product roadmap*. The former helps you capture the product vision and strategy; the latter helps you communicate a product roadmap that is based on product goals and outcomes. Note that you don't have to use the templates to take advantage of the model in Figure 1. You can happily use alternatives, provided that you clearly state the needs and business goals in your strategy and that you work with an outcome-based, goal-oriented roadmap.

Integrating Different Plans and Planning Levels

Having a product strategy and roadmap is great. But how do you ensure that the two plans are systematically connected with each other as well as the product vision and the product backlog? The model in Figure 1 achieves this in the following way: The vision guides the strategy; the strategy forms the basis for creating an effective roadmap; and the roadmap finally directs the product backlog.

With each step, the decisions you take become more specific; and higher-level goals are translated into increasingly detailed and focused ones. The vision is transformed into needs and business goals. From these, product goals are derived, which, in turn, guide the discovery of the right product backlog items. Additionally, the time frames become progressively shorter—from five to ten years covered by the vision to a product backlog that contains items for the next few months.

While the model suggests that product planning starts with vision setting, the connections between its elements are *bidirectional*. This means that bigger product backlog changes can trigger product roadmap modifications. For example, one or more goals on the roadmap might have to be adjusted, or the dates and time frames might have to be corrected. Similarly, larger product roadmap updates can lead to a product strategy change. You might find, for instance, that one of the business goals is unrealistic, that the needs have to be adjusted, or that a standout feature has to be reworked. Finally, if you can't find a validated product strategy, then you'll have to change or abandon the product vision.

Strategy and execution are therefore systematically connected in the model in Figure 1. Strategic decisions guide the implementation of product backlog items, and insights from the tactical work lead to changes in the product roadmap and strategy. This ensures consistent decisions, and it avoids a *strategy-execution chasm* where strategic and tactical

decisions are disjointed. In the worst case, such a chasm results in the development teams doing a great job at building the wrong product. Figure 2 illustrates the connection between strategy and execution.

In Figure 2, the strategic work guides the tactical product decisions, and insights from building software help you adapt the product strategy and roadmap. A validated product strategy directs the product roadmap, which helps discover the right product backlog items. These are transformed into product increments and eventually a product. The former allows you to determine the development progress using, for example, a release burndown chart.[3] The latter enables you to measure the product performance using key performance indicators (KPIs) like engagement, revenue, and customer satisfaction. The data you collect helps you inspect the strategy and roadmap and adapt and evolve the plans.[4]

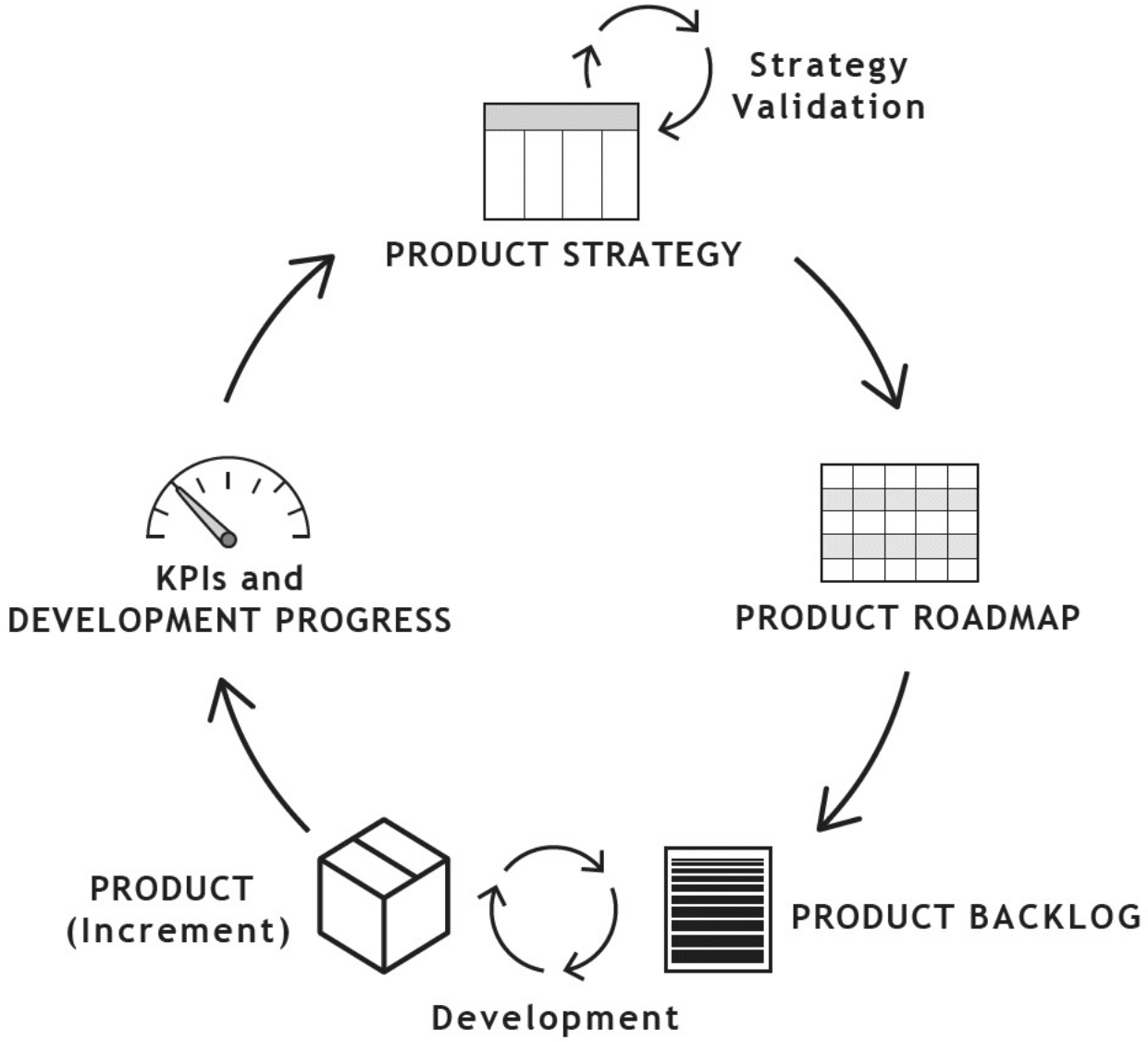


Figure 2: The Product Strategy Cycle

Note that there are also two smaller circles in the diagram: a validation one that indicates that the product strategy is best tested in an iterative fashion, a development one showing that a digital product may be developed using sprints.

Strategic Planning, Agility, and OKRs

When I explain the model in Figure 1, I am often asked two questions: Does it support an agile way of working? And can it be used with objectives and key results (OKRs)? The short answer to both questions is *yes*. But let me share the long answers with you.

The model facilitates agility in two ways: First, the product strategy and roadmap state higher-level goals that guide the detailed product decisions without anticipating or prescribing them. Instead, the product functionality and detailed user experience design are discovered while the product is developed using an agile framework like Scrum. Second, the product strategy and roadmap are regularly reviewed and updated. They are not static, fixed plans, but they evolve based on changing market conditions, technologies, and regulatory requirements. Additionally, insights from demoing and releasing product increments to users and customers are integrated into the strategy and roadmap, as Figure 2 illustrates. As mentioned above, product backlog changes can trigger a product roadmap update, which, in turn, might lead to a product strategy change. In other words, the strategic planning approach I suggest supports empirical management and adaptive planning.

If you want to use OKRs, then you can view the goals in Figure 1 as objectives. There are four goals: *vision*, user and customer goals captured as *needs*, *business goals*, and *product goals*. These goals form a cascading chain: The vision guides the needs and business goals, and the latter direct the product goals.[5] You might, for example, formulate the product goals as objectives and the corresponding dates, metrics, and features as the key results. But I personally find OKRs not well suited to manage products and set product-specific goals. Here is why: Using OKRs tends to lead to a rather text-heavy approach. It can result in documents that are comparatively lengthy and hard to understand. This feels like turning back the clock to the 1970ies when Andy Grove invented OKRs at Intel. It ignores the progress made in recent years by developing dedicated visual product management tools like my product vision board and GO product roadmap or Alexander Osterwalder's business model canvas (Osterwalder and Pigneur 2014).

Empowerment, Collaboration, and Strategic Decisions

Understanding what the product strategy and roadmap are and how they relate is important to effectively use the plans, align the stakeholders, and direct the work of the development teams. But it is not enough. Without the right level of empowerment, you risk that your strategy and roadmap are driven by senior stakeholders and that the plans primarily cater for their needs rather than the users' and customers'. But such an approach is hardly a recipe for achieving product success. In the worst case you end up with a *Frankenstein product*—a product that consists of an odd collection of features, has a weak value proposition, and offers a poor user experience. As the person in charge of

the product, *you* should own the product strategy and the product roadmap, and you should be empowered to have the final say on strategic decisions.

But this does not mean that you should create the strategy and roadmap on your own, hand the finished plans to the stakeholders and development teams, and expect them to put them into action. No matter how well thought-out your product strategy and roadmap are, they are worthless if the stakeholders and development teams do not buy into them. You should therefore involve them in creating and updating the plans, preferably in the form of collaborative workshops, as I explain in more detail later in this book. A collaborative approach offers the following three benefits:

First, it leads to better decisions, as it allows you to leverage the expertise of the stakeholders and development team members. Second, it creates transparency and a shared understanding—people know what needs to be done and why a decision was made. Third, it achieves greater buy-in. If you give people the opportunity to contribute to strategic decisions, they are more likely to support and implement them.

When you decide together with the stakeholders and development teams, cultivate an open mind, and don't cling to preconceived ideas. Instead, attentively listen to the individuals, and appreciate their ideas and concerns if even you disagree. But don't make the mistake of trying to please people or allow individuals to dominate. Instead, search for a decision that maximises the value your product creates and that attracts as much support as possible.

Overcoming a Lack of Empowerment

If you currently don't own the product strategy and roadmap and if you are not empowered to make strategic product decisions, then follow these three tips: First, increase your product strategy and roadmap expertise. This includes being able to create and validate a product strategy and derive a product roadmap from it—something this book will teach you. Otherwise, it will be hard for you to earn the stakeholders' trust and guide them. Second, strengthen your leadership skills and your ability to influence others. This includes practising active listening and constructively dealing with disagreements and conflicts, as I explain in more detail in my book How to Lead in Product Management (Pichler 2020). Third, lobby for more authority and engage with the senior decision-makers in the organisation. Explain to the individuals why empowerment is a perquisite for achieving product success. Your Scrum Master or agile coach may be able to help you with this, as facilitating organisational change is part of their job. This includes establishing an effective product management function.

Four Product Success Factors

The ultimate goal of creating a product strategy and roadmap is to achieve *sustained* product success. You want to ensure that your product does a great job for its users and customers and that it generates value for the business—on a continuous basis. While realising product success cannot be reduced to a simple formula, there are four factors that have a profound impact on it. These are: desirability, feasibility, viability, and ethicality, as shown in Figure 3.

Desirability is probably the most important factor. If people don't need or want to use a product, it will fail. To be desirable, a product must solve a specific problem or offer a tangible benefit. An example of the former are Apple's AirTags, which help people find

their keys and other misplaced items. Contrast this with Sonos, a wireless hi-fi system that allows people to enjoy music by providing easy access to a range of streaming services. But no matter if a product solves a problem or provides a benefit, it must create value for its users—or it is doomed.

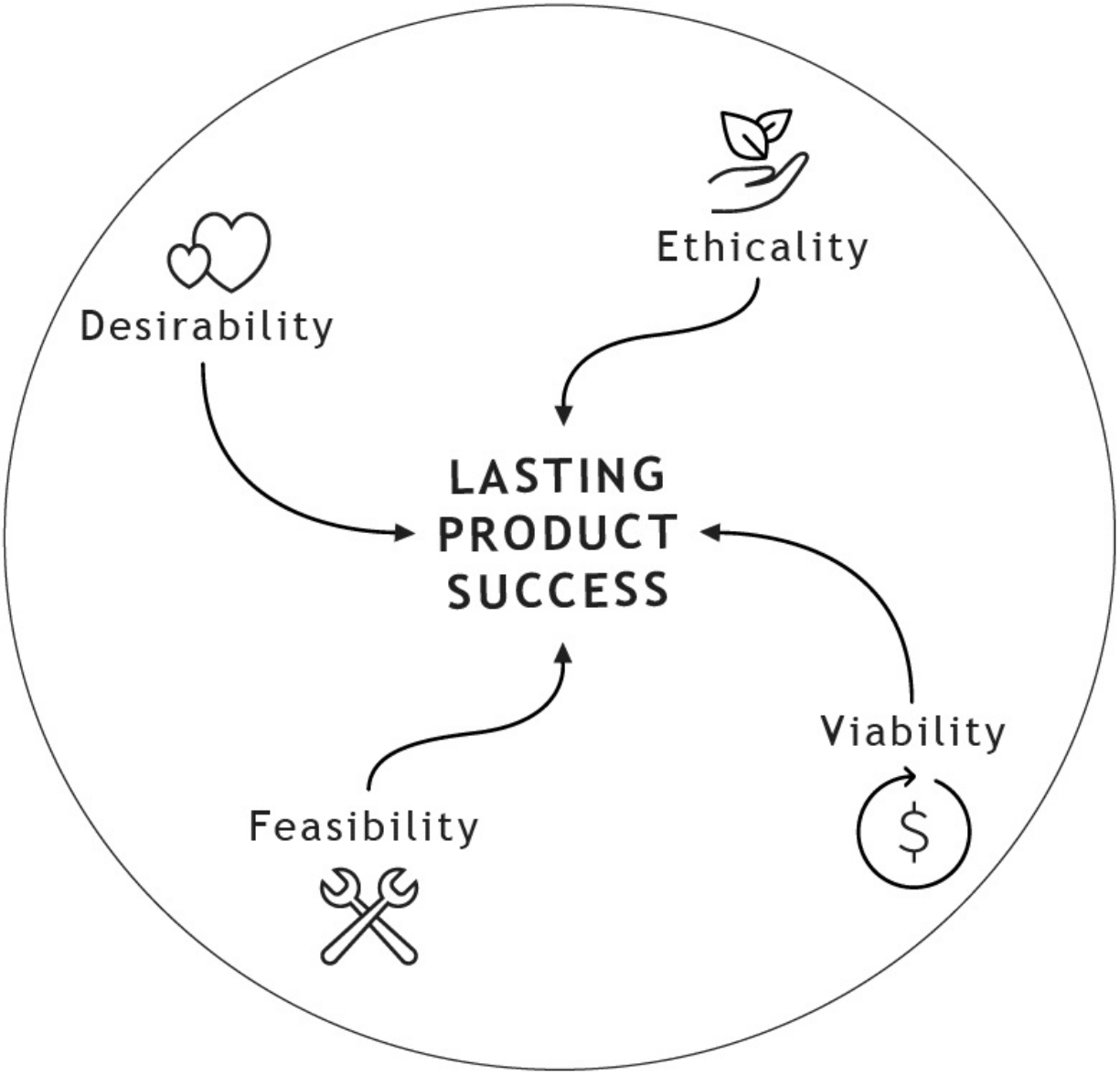


Figure 3: The Four Product Success Factors

Feasibility implies that it is possible to design and build the product; the technologies required exist or can be developed. Additionally, enough people with the right skills are available or can be recruited. If, for instance, developing the product requires advanced machine-learning algorithms, then you'd have to ensure that appropriate machine-learning frameworks exist or that they can be created.

Viability means that developing and providing the product is viable from a business perspective. The product therefore must create enough business benefits to justify spending money on it. These may include generating revenue, reducing cost, increasing productivity, and strengthening the brand. Examples of revenue-generating products are Google Search, which makes money through ads, and Microsoft Word, which is monetised through subscriptions (as part of the Office suite). Contrast them with products like the Google Chrome browser and Microsoft Edge, which offer different business benefits. They allow their companies to control how users access the Internet, tie them to their respective ecosystems, and make it more likely that they interact with revenue-generating offerings.

Ethicality, finally, states that a product must not cause any harm to people and the planet. It must not affect people's mental wellbeing, for instance, by getting them hooked or by offering content that promotes misinformation, self-harm, or violence. Additionally, an ethical product does not contribute to climate change and does not damage the environment by how it is developed, provided, and if it includes hardware, manufactured, delivered, and disposed of. To achieve this, you will benefit from a business model that is fair to all parties and from making ethically sound design and programming decisions, for example, by avoiding the use of dark patterns and mitigating machine learning biases, as I explain in more detail later in this book.

A Brief History of the Four Success Factors

The first three factors—desirability, feasibility, and viability—were originally suggested by Tim Brown in his book *Change by Design*. He describes them as competing constraints that must be balanced to innovate successfully. But as Brown notes, this could result in "dreaming up alluring but essentially meaningless products destined for the local landfill—persuading people … 'to buy things they don't need with money they don't have to impress neighbours who don't care.'" (Brown 2009, pp. 20) What's more, the impact digital products can have on people's mental health makes it compulsory in my mind to add ethicality as a fourth success factor. On the positive side, an ethical product increases the chances of achieving lasting product success, and it reduces the risk that the reputation of the company suffers.

A Product Strategy Process

Making the right strategic product decisions usually requires you to collect data and acquire new knowledge. You might have to talk to prospective users, perform a competitive analysis, and investigate suitable technologies—to name just a few common tasks. This work needs to be planned and managed, and to do this, you will benefit from establishing the right process. This process comes in two flavours, timeboxed and continuous strategizing.

Timeboxed Strategizing

Whenever you develop a brand-new product or make a bigger change to an existing one, I recommend using an innovation process like the one shown in Figure 4.

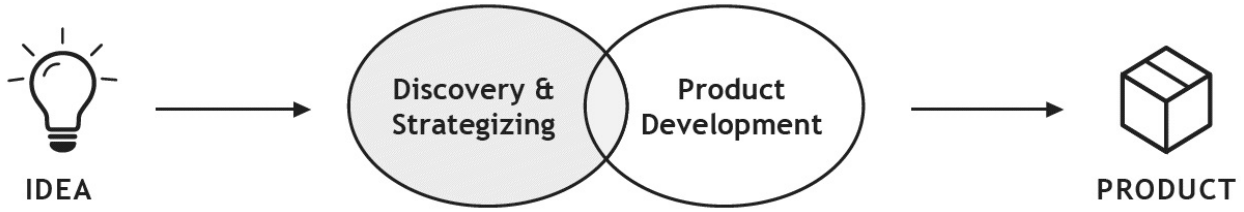


Figure 4: Innovation Process with Timeboxed Strategizing

The process in Figure 4 consists of two sets of activities, product discovery/strategizing or and product development.[6] The first set determines if and why the product should be developed. This might require observing and interviewing users, developing and testing prototypes, and exploring the product's underlying business model. The results include a validated product strategy, an actionable product roadmap, and an initial product backlog, as well as a valid business model, high-level user experience (UX) design concept and coarse-grained software architecture.

The second set of activities determines what the product should look like and do and how the product should be developed. This involves discovering and implementing the right UX design and functionality as well as making the right technical decisions. If you use a Scrum-based development process, you will test product increments with selected users early and often, for example, by demoing and releasing the software to them. This helps you learn from their feedback, adapt the product while it is being developed, and maximise the chances of creating a product that does a great job for its users.

Note that Figure 4 shows discovery/strategizing and development as overlapping activities rather than distinct phases. The reason: Some development tasks like high-level user experience and architecture design are often part of the initial work. This allows you to address risks that are related to the desirability and feasibility of the product. Additionally, it sets up the development work and removes the need for a preparation sprint or sprint 0. Be careful, though, not to make the mistake of carrying out big design upfront (BDUF). Only address the major UX and architecture risks and make most of the design and implementation decisions when the actual product is developed.

As it's hard to correctly predict how much time will be required to complete the necessary work and create a validated product strategy and actionable roadmap, I recommend using a time box, a fixed period during which the work has to be completed. Your goal should be to carry out *just enough* product discovery and strategizing work to create the plans and be able to make an informed go/no-go decision about developing the actual product. If you are unsure how big the time box should be, then start with one month and hold weekly review meetings where you assess the progress, as I discuss in more detail in the chapter *Product Strategy Validation*.

Finally, don't forget to involve the key stakeholders and development team members in the strategizing work. While it can be helpful that the entire development team participates in the activities, it is often sufficient to ask the team to nominate three people: a member with the right UX skills—assuming that the product is end-user facing, a member with the relevant architecture and technologies expertise, and a member who has the right quality assurance capabilities. This will enable you to address the key user

experience and technology risks, including having the right development and test tools available.

Design Thinking and Timeboxed Strategy Work

If you are interested in design thinking, you might be wondering if you can use design-thinking concepts to carry out the strategizing work shown in Figure 4. Before I answer this question, let's briefly explore what design thinking is. The approach can be described as five sets of activities, which are commonly called *empathise*, *define*, *ideate*, *prototype*, and *test*.

The first set helps you understand who your users are and what they need, for instance, by observing and interviewing them. The second set defines the specific user problem that you want to solve. The third set generates ideas for possible solutions, which are turned into simple prototypes in the fourth stage. The final set tests the prototypes with selected users to find an approach that is likely to result in a successful product.

Note that the activities are usually not carried out sequentially. Instead, you may find yourself looping back to a previous stage. For example, you might discover in the prototype stage that none of the solution ideas receive positive feedback from prospective users. Consequently, you'd have to go back to the ideate stage and come up with new ones.

Can you use the design-thinking stages to structure and organise the strategizing work shown in Figure 4? Yes, you can. What's more, the stages are compatible with the approach of creating and validating a product strategy discussed in this book. You should therefore be able to apply the practices in a design-thinking context.

Continuous Strategizing

As helpful as timeboxed strategizing is to create a new product strategy and roadmap, it is not enough. The world doesn't stand still; markets and technologies change; new competitors emerge. It is therefore important that you practise continuous strategizing and regularly review and adapt your strategy and roadmap. This includes the following five activities: First, determining the product performance using key performance indicators (KPIs), such as engagement, conversion, and customer satisfaction. Second, reviewing market trends that might impact your product like machine learning and Internet of Things (IoT). Third, keeping an eye on the competition and monitoring the changes you competitors make to their product portfolios and individual offerings. Fourth, following developments at your own company that might affect the product strategy and roadmap, for instance, a change in the business strategy. Fifth, meeting selected users and customers at least once per quarter, be it online or onsite. This will help you empathise with them and discover new ideas for product enhancements and new offerings.

You can think of continuous strategizing as an ongoing workflow. This workflow directs the tactical work and informs the detailed product decisions that are captured in the product backlog.[7] At the same time, insights from the tactical work influence the strategic decisions and help you adapt the product roadmap and strategy. As mentioned before, the strategic and tactical work should be connected, and strategic and tactical product decisions should be closely aligned.[8]

To help you effectively carry out the continuous strategizing work, I recommend the following two measures: First, make sure that you allocate enough time, at least half a day per week as a rule of thumb. Some product people like to spend one hour per day on

continuous strategizing, others prefer to carry out the necessary work once or twice per week. Note that the amount of uncertainty and change present will impact the strategizing effort: A young product requires a higher effort compared to an older, mature one. Second, schedule regular strategy and roadmap reviews—roughly once per quarter—and invite stakeholders and development team members to them.

The first measure helps you avoid nasty surprises such as a competitor offering a new killer feature. It makes it more likely that you notice early warning signs like declining sign-up rates, increasing churn, or growing number of support calls. This allows you to be responsive and take action early so you don't end up firefighting and possibly having to resolve a crisis. The quarterly reviews help you consider longer time frames and bigger trends. By involving stakeholders and development team representatives—ideally the same people who helped you create the current product strategy and roadmap—you leverage people's collective expertise, create alignment, and secure buy-in. While I do recommend that you schedule the reviews in advance, you should, of course, not wait for the next review if there are new developments that need to be urgently addressed. Instead, hold the next collaborative review as soon as possible.

Tips for Making Time for Continuous Strategizing

As the person in charge of the product, you might find that your workload is already so high that practising continuous strategizing seems impossible. If that's the case, then follow these four tips to free up time:

First, don't carry out tasks that are not part of your actual role. This includes taking on Scrum Master duties, which is a common mistake in my experience.[9] Second, delegate and share some of the work. For instance, the development team members might be able to do some product backlog refinement work on their own. Third, don't task-switch. Instead, do one thing at a time. Every time you start a new task, be it reading an email or looking at new data, you need time to remind yourself of what has to be done and how to do it. Fourth, take regular breaks that allow you to recharge. Working too hard for too long is counterproductive. Productivity drops and mistakes increase.

Be aware that de-prioritising or neglecting strategic work will most likely lead to unplanned work in the future. If you don't make time to listen to the users and watch the market, then don't be surprised if engagement and referral rate are down or if a competitor leapfrogs you. In this sense, strategizing is like riding a bicycle or driving a car: If you don't look ahead, you're likely to crash.

Digital Products and Product Strategy

A product strategy and a product roadmap are plans that describe the value a *product* should create. But what is a product? While this might seem a trivial question, I find that many organisations lack a clear and shared answer to it, especially when it comes to digital products. This, however, can cause the following two issues:

First, you might mistake a product part—like a feature or component—for the actual product. I view a feature as a product capability, as a large piece of functionality, for instance, the ability to save a file. A component is an architecture building block like a service, layer, or subsystem. It might be the code that handles data storage, to stay with the example. This mistake would cause you to create a feature or component strategy and a feature or component roadmap. But these plans would be too detailed and fine grained. They would provide no real value despite increasing the planning and coordination effort.

Second, you might make the mistake to view a product portfolio or a product bundle like Adobe Creative Cloud and Microsoft Office as a product.[10] This can lead to a strategy and a roadmap that is too coarse-grained and high-level. Consequently, it does not offer enough guidance for the stakeholders and development teams working on the individual products.

These mistakes show that it's important to be clear on what a product is and if the asset you manage is a product before you create a product strategy and roadmap. I view a product as an entity that creates specific value for a group of people, for the users and customers, and for the organisation that develops it. The former is achieved by solving a problem—think of Google Search, which addresses the challenge of finding information on the Internet—or by providing a tangible benefit—take Microsoft Word, which allows people to easily create and edit documents. Creating value for the business is accomplished by directly generating revenue, like Microsoft Office does, helping promote or sell other products or services, think of an online store like Amazon.com, and increasing productivity and reducing cost, as an internal software platform does, for instance.[11] To put it differently, a product is a value-creating vehicle: It's an asset that exists to enhance people's lives and to help an organisation grow and prosper.

If an asset does not create value for its users, customers, and the company, then it is not a product. Take Microsoft Word as an example. As a user, I can open a file, edit it, and save it. These three capabilities represent steps in a common user journey, they might be developed by dedicated teams, and they might require complex technical solutions. But I would view them as features, not products, as they do not provide value to the users and to Microsoft on their own.

Tips for Identifying Digital Products

If you struggle to determine which assets are products and which ones aren't, then follow this approach: Start with the users and customers and identify your products *outside-in*. Ask yourself why people use the capabilities you offer. Which problems do they try to address, which benefits do they want to obtain, or which jobs do they try to get done? You can answer these questions by observing and interviewing users and by using relevant analytics data.

Additionally, begin with revenue-generating products, which are usually comparatively easy to identify: These are the assets people are willing to pay for. An example would be the products sold on an online retailer's website. Then consider the supporting products like the website and mobile apps that allow the customers to find and purchase the right products. Finally, determine any internal technical products like a software platform or a data repository. This should give you a clear understanding of the products in your business and put you in a good position to make the right strategic product decisions.

PRODUCT STRATEGY FOUNDATIONS

Without a solid foundation, you'll have trouble building anything of value.
Erika Oppenheimer

As its name suggests, this chapter lays the basis for developing an effective product strategy—it contains essential strategy concepts and techniques. It will help readers who are new to the topic get up to speed. For seasoned strategy practitioners, it provides the opportunity to brush up their knowledge and reflect on their practice. Let's start by discussing what exactly a product strategy is.

Understand What a Product Strategy Is

What do using Google Search and booking a car on Uber have in common? Both are everyday technology experiences that require well-designed products. These have to be able to handle varying loads, process complex interactions, and manage huge amounts of data. To achieve this, requirements have to be captured, design sketches have to be created, and architecture and technology decisions have to be made. While attention to the details is necessary to create a successful product, making tactical decisions without being clear on the overall strategy is like playing Russian roulette: If you are lucky, you'll make the right choices. But chances are that you are not. This is where the product strategy comes in. It paints the big picture, captures the strategic product decisions, and guides the execution.

The Product Strategy Defined

A product strategy is a high-level plan that helps you realise your vision and states how you intend to achieve product success. It explains who the product is for, and why people would want to use and buy it; what the product is, and what makes it stand out; and what its business goals are. In other words, an effective product strategy captures the market and needs, the key features or differentiators, and the business goals. Figure 5 illustrates these elements.

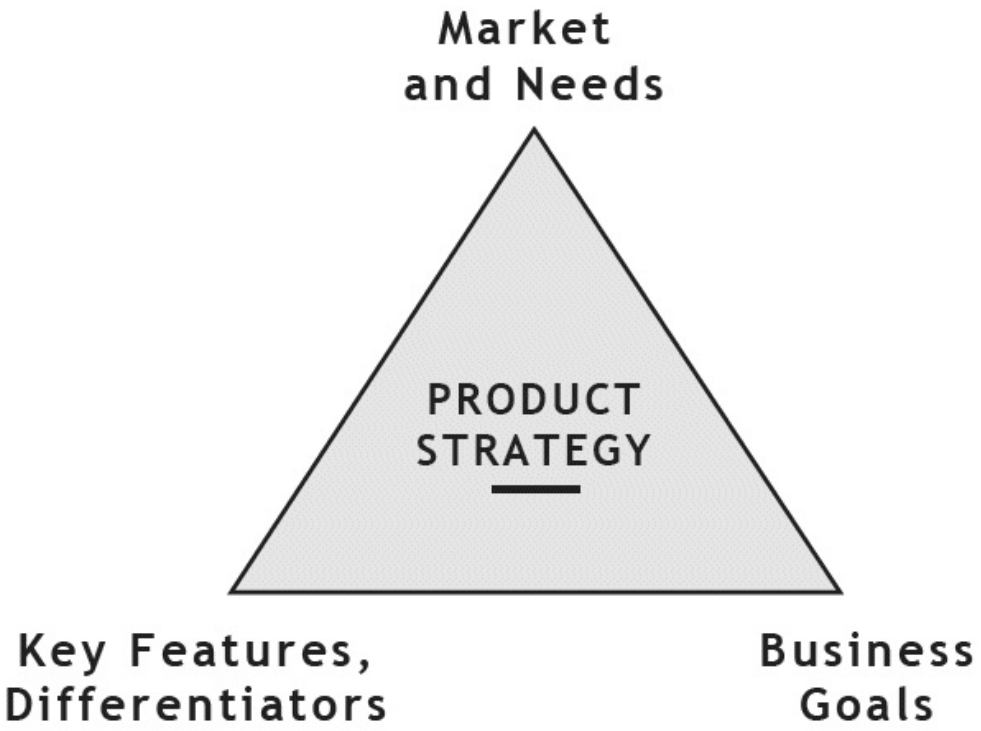


Figure 5: The Elements of the Product Strategy

The *market* in Figure 5 describes the target users and customers of your product, the people who are likely to use and buy it. The needs capture the reason why people would want to interact with and purchase the product. A need is a specific problem that people want to see addressed, a tangible benefit they would like to obtain, an outcome they want to achieve, or a job that they want to get done. Think of a product like Google, which solves the problem of finding information on the Internet, compared with a product like LinkedIn, which provides the benefit of connecting with colleagues and other professionals.

The *key features* are those aspects of your product that are crucial to creating value for the users and customers and that encourage people to choose it over competing offerings. Take, for instance, the first iPhone and its key features of mobile Internet, an iPod-like digital-music player, and a touch screen; or the Google Chrome browser with its focus on speed, safety, and simplicity. As these two examples show, the point is not to list all product features in your strategy—that's done in the product backlog—but to focus on the three to five features that influence a person's decision to buy and use the product.[12]

The *business goals* capture how your product is going to benefit your company, and why it is worthwhile for the business to invest in the product. Is it going to generate revenue, help sell another product or service, reduce costs, or increase brand equity? Take the iPhone and the Google Chrome browser mentioned earlier. While the iPhone generates a large portion of Apple's revenue at the time of writing, the Chrome browser does not earn any money for Google. But it does allow the company to control the way

people access the Internet, and it has reduced Google's dependency on third-party browsers such as Mozilla Firefox and Microsoft Edge.

Note that an effective product strategy requires making tough decisions. By selecting a specific group of users and a main need, for example, you decide *not* to address the needs of other groups of people. You essentially say no to other markets or market segments and to other needs—at least for the time being. Additionally, don't think of the product strategy as a fixed plan that is once created and then simply executed. Instead, your strategy will change as your product grows and eventually matures. Consequently, you will have to regularly review and adapt the product strategy, at least once a quarter as a rule of thumb.

Bootstrapping a Product Strategy

Knowing which elements an effective product strategy should contain is all good and well. But how do you come up with the plan in the first place? To answer this question, let's look at three scenarios: creating a brand-new product, making a bigger change to an existing offering, and incrementally enhancing a product.

If you are about to create a brand-new product, then do just enough initial discovery work to be able to create an initial product strategy. This may include observing target users, interviewing people, and carrying out some competitive analysis—all of which I'll discuss in more detail later in this book. Your initial strategy does not have to be perfect. But it must be good enough so you can spot and test the assumptions it contains.

If you manage an existing product and plan to make a bigger change to it, for instance, to take it to a new market, then start by capturing its current strategy. Next determine which adjustments are necessary to support the desired change and create a new product strategy. This might also require some upfront discovery work, along the lines described above. Once the new strategy is available, validate it and address the risks and assumptions it contains.

Finally, if you manage a stable product that is incrementally enhanced but lacks a clear product strategy, then capture its current market, needs, standout features, and business goals. Next, assess if the plan will help you maximise the value your product creates in the future. If that's not the case, adapt it. This might require a smaller adjustment like modifying a standout feature or business goal, for example. But it might also necessitate a bigger change.

Don't forget to involve the stakeholders and development team members in the strategizing work. This allows you to leverage their expertise and make the right decisions; it creates a shared understanding and avoids diverging interpretations; and it increases the likelihood that people will implement the strategy.

Think Big and Describe Your Vision

As the product strategy describes how you intend to realise your vision, it is helpful to begin by capturing this overarching goal. I view the product vision as the ultimate reason for creating a product, as an inspirational goal that describes the positive change the

product should bring about. Say I want to create an app that helps people become aware of what, when, and how much they eat. My vision, then, might be to "help people eat healthily;" and the strategy might be to create an app that monitors their food intake in conjunction with a smart watch, fitness band, and smart food scales.

An effective product vision that motivates and guides people has the following six qualities:

1. *Inspiring*: An inspiring vision creates a meaningful purpose for everyone involved in making the product a success including the stakeholders and development team members. It helps people understand how their work relates to a bigger whole and how their efforts create a positive change. It also allows you, as the person in charge of the product, to understand if dedicating your time and energy to the offering is worthwhile and sustainable. If the vision resonates with you, then this will help you do a great job, especially when the going gets tough. If that's not the case, then you should maybe look for a different challenge. Life is too short to work on products you don't believe in.
2. *Shared*: A shared vision unites people. It acts as the product's true north, creates alignment, and facilitates collaboration. A vision is shared when the key stakeholders and development team members support the goal and are happy to work towards it. If that's not the case, then it will be difficult to encourage the individuals to support the product strategy and roadmap and put them into action.
3. *Ethical*: An ethical vision gives rise to a product that benefits its users and customers and that does not cause any harm to people and planet. While the vision alone doesn't achieve ethicality, it plays a crucial role, as it describes the intention for offering the product. You ultimately have to ask yourself why you want to provide the product. Is it to help others, or is it to maximise your own benefits?
4. *Concise*: A concise product vision is easy to communicate, understand, and remember. To achieve this, I like to capture the vision as a brief statement or a slogan—a short, catchy phrase such as "help people eat healthily."
5. *Ambitious*: You can think of the product vision as a big, hairy, audacious goal, abbreviated as BHAG (Collins 2005). Such a vision increases the chances of providing a continuous purpose in an ever-changing world, compared to a narrow, specific one. I would therefore choose a big, ambitious vision like "help people eat healthily" over a more specific, narrower one like "help people lose weight." Note that a big, ambitious product vision is not measurable. It truly is an inspirational, grand goal.
6. *Enduring*: Despite its name, a product vision should not describe the product or solution. For example, "offer a weight loss mobile app" and "become the number one weight loss app provider" are not effective visions. Instead of referring to the product, state the positive change you want to bring about, such as "healthy eating." This allows you to change the product strategy but keep your vision stable. Say that it turns out that my idea of developing a healthy-eating app is ill

conceived. With a vision like "healthy eating," I can explore alternatives, for example, writing a book on healthy eating. A vision that is not tied to a product can therefore provide direction for an extended period—the next ten years as a rule of thumb—during which the product strategy and the product change and evolve.

A great way to ensure that your vision exhibits the qualities above is to create it together with the key stakeholders and dev team members in a collaborative workshop. Invite the right people to a joint session, be it online or onsite, and encourage the participants to describe the purpose that they associate with the product. Then look for a product vision that everyone can support.

As the vision is truly fundamental, you should take the time required to create an inclusive vision that resonates with everyone. Resist the temptation to shortcut or rush the decision-making process. Similarly, don't allow the most powerful person to dominate and don't agree on the smallest common denominator. Otherwise, you'll end up with an ineffective vision that fails to inspire and guide people's work.

Does Every Product Require its Own Vision?

Every product should have a vision, but not every product requires its own, unique one. Say your product is part of a group of related products like Microsoft Office. In such a case, I recommend using one overarching vision for all products. In Microsoft Office's case, this might be, "help people collaborate in real time." Word, PowerPoint, Excel, and the other Office members would then share this vision.

Use the Business Strategy to Guide Strategic Product Decisions

A product is a means to an end. By benefiting its users and customers, it should create value for your company and help move it in the right direction. It is therefore important that your product strategy supports the overall business strategy. An effective business strategy describes how the company intends to be successful. Answering the following five questions—which are based on Martin (2013)—will help you come up with such a strategy.

1. *What is your winning aspiration?* What is the company vision? State the purpose of the organisation that helps identify the right strategic objectives. Take, for instance, Tesla's goal "to accelerate the world's transition to sustainable energy."[13]
2. *Where will you play?* Clearly describe the areas in which the company will compete to fulfil its aspiration. Who should benefit from your offerings? Do you intend, for example, to address existing markets, or do you aim to create new markets? Which geographies and regions do you want to serve?
3. *How will you win?* What is your competitive advantage? For instance, do you want to exercise cost leadership and offer products at low prices? Do you want to take advantage of differentiation and provide uniquely desirable products? Or do

you focus your efforts and address niche markets?[14] Answering these questions requires an understanding of the strengths and weaknesses of your company and the competition.

4. *What capabilities must be in place?* Which new products and services do you require? Which existing products should you enhance, and which offerings should you discontinue? Think of Apple's strategy change to earn more money with software and services and the launch of Apple TV+ and Fitness+ in 2019 and 2020 respectively.
5. *Which management systems are required?* Which processes and structures are necessary to build the appropriate capabilities and reinforce your organisation's strategic choices? This might involve creating or strengthening a product management organisation or investing in an agile transition.

A business strategy that answers the questions above provides the context for making the right strategic product decisions, for instance, to decide which products should be created and enhanced and which market a specific product should address. To put it differently, the business strategy should guide an individual product strategy. If your company does not have a business strategy, or if you are unaware of it, then consider delaying strategic product decisions until such a strategy has become available—unless you work for an early-stage start-up, in which case the two strategies may well be identical.

Business Vision and Product Vision

I like to recommend working with a separate business and product vision—unless the company is an early-stage start-up. The business vision should state the company's overarching aspiration while the product vision should capture the positive change a specific offering should create. The product vision, however, should be aligned with the business vision so that the purpose for creating the product supports the company vision.

Understand Your Product's Innovation Type

Products are value-creating vehicles. To generate value, a product has to offer something new; it has to innovate to a greater or lesser extent. Innovations range from small incremental steps, such as improving the user experience for an existing product, to big and bold ones—think of the original iPhone, the Nintendo Wii, and the Uber taxi service. It's important to understand which innovation strategy your product executes, and which innovation type it represents, as this will shape the product strategy and determine the environment you need to successfully carry out the strategizing work. A helpful way to classify innovations is the *Innovation Ambition Matrix* developed by Bansi Nagji and Geoff Tuff, which Figure 6 illustrates.[15]

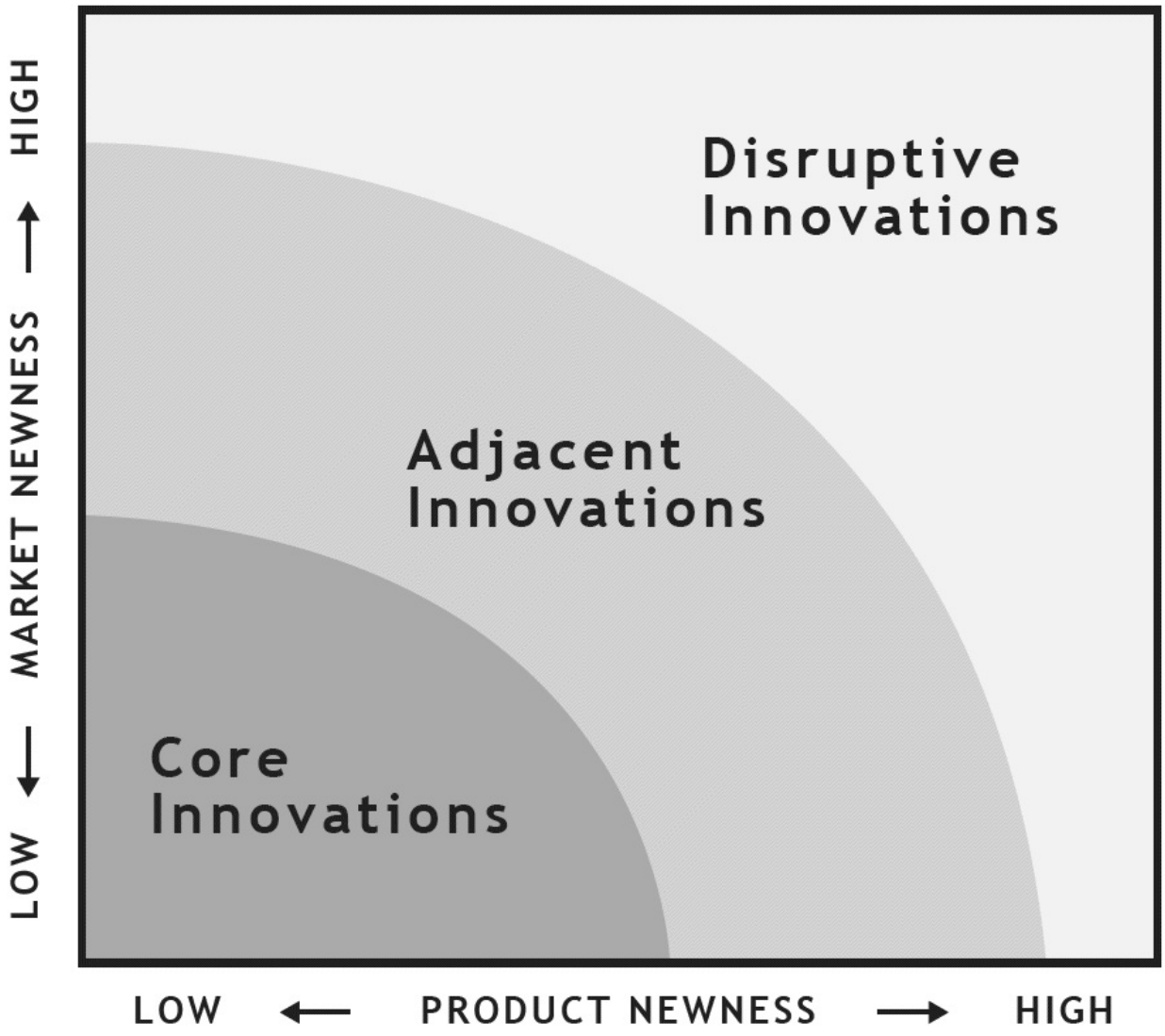


Figure 6: The Innovation Ambition Matrix

The matrix in Figure 6 considers the newness of the product on the horizontal axis and the newness of the market on the vertical axis. This allows us to distinguish three different innovation types: core, adjacent, and disruptive.[16] Let's discuss these innovation types by exploring their growth potential and risk, the ability to create a reliable business case, and the mindset and organisational context, which you will require.

Core Innovations

Core innovations optimise existing products for established markets. They draw on the skills and assets your company already has in place, and they make incremental changes to current products. These initiatives are core to your business, as they generate today's revenues. Most of your company's products are likely to belong to this category—unless you work for a start-up. Examples of core innovations include Microsoft Windows and Office. Both are major revenue sources for the company at the time of writing. The

longer-term growth potential of core products is low, and so is the amount of risk and uncertainty present.

Your ability to create a reliable financial forecast or business case is high due to your in-depth knowledge of the market and the product. Because core products leverage existing assets, a conservative attitude is appropriate. You should aim to protect the product, focus on operational excellence, avoid mistakes, optimise the existing business model, and use proven technologies—unless you decide to make a bigger change to your product, such as taking it to a new market, which would turn it into an adjacent innovation.

Adjacent Innovations

Adjacent innovations involve taking something your company does well into a new space—for instance, offering an existing product in a market that's new to the company or creating a new product for an existing market. Examples of the former include Microsoft entering the server market with Windows NT in 1993 and Facebook moving into the online payment space with its Messenger application.[17] Examples of the latter include the Apple TV and Google's Chrome browser. Both companies entered an existing market—TV set-top boxes and web browsers, respectively—with a new product.

Adjacent innovations allow you to create new revenue sources, but they require fresh insights into customer needs, demand trends, market structure, competitive dynamics, technologies, and other market variables. You may also have to acquire new skills, use new technologies, and adapt an existing business model. The amount of risk and uncertainty present is therefore considerably higher than in core innovations. It consequently requires more time to develop a valid product strategy, and it becomes difficult to create a reliable financial forecast.

To succeed with adjacent innovations, you should adopt an inquisitive attitude, be willing to take informed risks, and feel safe enough to make mistakes and fail. You will benefit from having a dedicated product team that is loosely coupled to the rest of the organisation and that applies agile product development practices.

Disruptive Innovations

Core and adjacent innovations provide you with the benefit of leveraging existing assets, both intellectual and material. This makes the challenge of innovating successfully manageable.[18] Unfortunately, such innovations also share a significant disadvantage: they address an existing market, and their growth prospects are limited by your ability to grow the market and capture more market share—that is, to attract more users and customers. To experience higher long-term growth, your company has to invest in disruptive innovations. Apple, for instance, disrupted the mobile-phone market with the first iPhone by offering a product with superior usability and mobile Internet; Nintendo disrupted the games-console market with its Wii, which could be used without a traditional control or keyboard and was offered at a comparatively low price; Amazon disrupted the retail book

market with its online store, making it easier and more convenient for consumers to shop, and offering a greater choice and lower prices.

While disruptive products often use disruptive technologies—for example, the touch screen in the case of the iPhone, and the Internet in the case of Amazon—a disruptive technology does not necessarily create a disruptive innovation. Instead, the latter typically solves a customer problem in a better, more convenient, or cheaper way than existing alternatives. A disruptive product also creates a new market by addressing nonconsumption: It attracts people who did not take advantage of similar products. But as the disruptive product matures, it makes inroads into an established market, reconstructs market boundaries, and disrupts the market. Take the iPhone as an example. The established players, including Nokia and BlackBerry, did not perceive the original iPhone as a threat; its business features, such as email integration, were too weak. But as the iPhone improved and offered an increasing range of business and productivity apps, more and more people began to use the product, and the market share of Nokia and BlackBerry phones started to decline. The first iPhone also removed the traditional distinction between business and consumer segments, thereby changing the market boundaries.

As crucial as disruptive products are for enabling future growth and securing the long-term prosperity of your business, most established companies struggle to leverage such innovations effectively. To achieve disruption and to do different things, a company must do things differently and therefore disrupt itself—at least to a certain extent. It has to discontinue some of the practices that have helped it become successful, acquire new skills, find new business models, and often embrace—and in some cases develop—new technologies, such as the touch screen for the iPhone and the motion controller for the Wii. The effort to create a valid product strategy is significantly higher than for adjacent innovations. It may take you several months to find a strategy that is likely to result in a beneficial, technically feasible, economically viable, and ethical product.

Succeeding with disruptive innovations requires an entrepreneurial mind-set and the ability to experiment, to make mistakes, and to fail. Using an *incubator* can help you with this, as I discuss in more detail in the chapter *Product Strategy Validation*.[19] Additionally, you will benefit from having a small team with full-time members and employing agile development practices. Be aware that creating a reliable financial forecast is impossible for disruptive innovations. As Clayton Christensen writes in his book *The Innovator's Dilemma*, "markets that don't exist can't be analysed."[20] Requiring a solid business case can, in fact, prevent you from creating disruptive products. It's often better to use the risk of inaction—the danger of not investing in a disruptive product and therefore losing out on future revenue and profits—to justify the investment and acquire a budget.[21]

Summary

Table 1 summarises the three innovation types. It also shows that you should adopt different practices and manage products differently depending on their innovation type.

Note that over time, successful disruptive and adjacent products turn into core ones. A good example is the iPhone. While the first version was a disruptive innovation, it has

become a major revenue source for Apple. But you can also move a core product into the adjacent space by taking it to a new market. Think of the iPhone 5C, for instance, which was aimed at a younger audience and emergent markets. The bottom line is: To grow organically, companies have to continually look for new growth opportunities and invest in adjacent and disruptive products—the products that generate tomorrow's cash.

Table 1: The Three Innovation Types and Their Impact

Areas	Core Innovation	Adjacent Innovation	Disruptive Innovation
Product	Optimise an existing product for an established market.	Create a new product for an existing market or take an existing product to a market that's new to the company.	Create both a new product and a new market.
Growth Potential and Risk	Low	Medium	High
Attitude	Conservative—protect existing assets, focus on operational excellence; avoid mistakes; optimise existing business models.	Inquisitive—take informed risks; look for new growth opportunities while leveraging existing skills, assets, and business models.	Entrepreneurial—create new assets, develop new skills, and find a valid business model. Mistakes and failure are unavoidable.
Organisation	Business as usual; matrix organisation.	Dedicated product team that is loosely coupled to the rest of the organisation.	Incubator with a small, full-time product team that is autonomous and empowered.
Technologies	Proven technologies; changes usually result in incremental improvements.	New technologies may be necessary to gain a competitive advantage.	New, disruptive technologies are likely to be required.
Discovery and Validation Effort	Low (hours to days)	Medium (weeks)	High (months)
Reliable Financial Forecast	Possible	Difficult to create	Impossible to create

Take Advantage of the Product Life Cycle Model

The product strategy aims to maximise the chances of achieving product success—to ensure that your product grows and prospers on a continuous basis. A helpful model to understand how products develop over time is the product life cycle. The idea behind this model is simple: Like a living being, a product is born or launched; it then develops, grows, and matures. At some point it declines, and eventually the product dies and is taken off the market, as Figure 7 shows.[22]

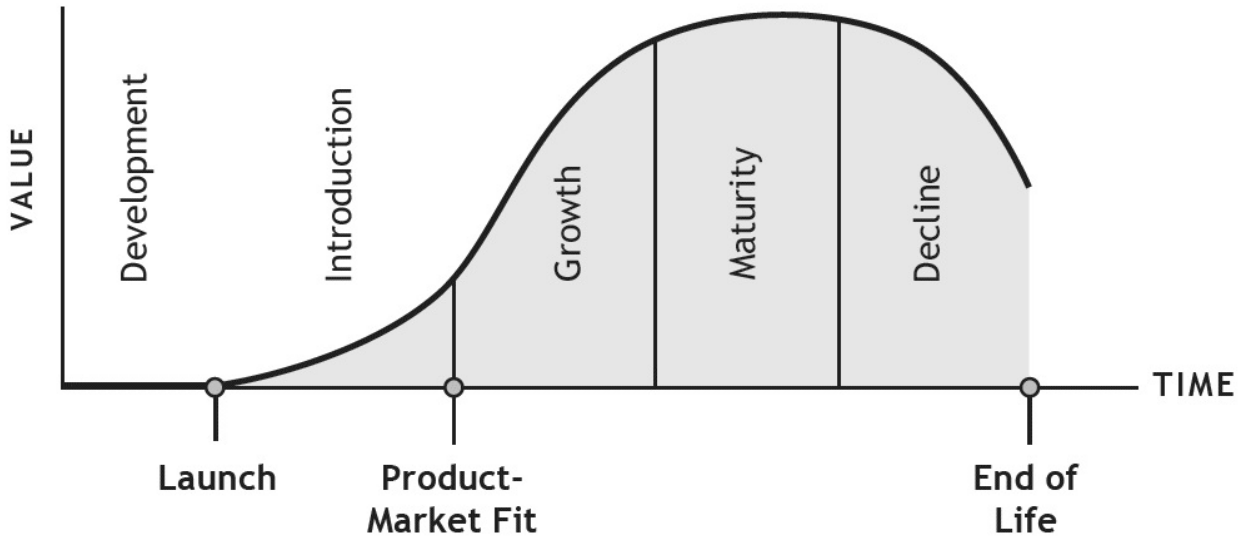


Figure 7: The Product Life Cycle Model

The product life cycle model in Figure 7 presents five stages: development, introduction, growth, maturity, and decline. Note that the first stage is not limited to development work in the narrow sense of the term. It also includes the necessary product discovery and strategizing work. In addition to the stages, Figure 7 contains three important events in the life of a product: launch, when the product first becomes available; achieving product-market fit (PMF), when your product is ready to serve the mainstream market;[23] and end of life, when you decide to discontinue your product. Of the five stages, growth and maturity are the most attractive ones, as they provide you with the biggest business benefits. A revenue-generating product should become profitable around PMF, and it should offer the highest profit margin in maturity.

While the curve in Figure 7 is roughly bell-shaped, your product's actual trajectory may differ significantly: It may be steeper or flatter. This demonstrates that the life cycle model is not a predictive tool that forecasts the value your product will generate. Instead, it is a sense-making model that helps you reflect on how your product is doing so you can make the right strategic decisions. To leverage the model, you have to first define the value your product creates and then track it over time. The former is done by creating and validating a product strategy; the latter is achieved by using the appropriate key performance indicators.

To see how the product life cycle model can be applied, let's look at Apple iPods. Figure 8 illustrates the life cycle of the iPod family by showing iPod sales per year.

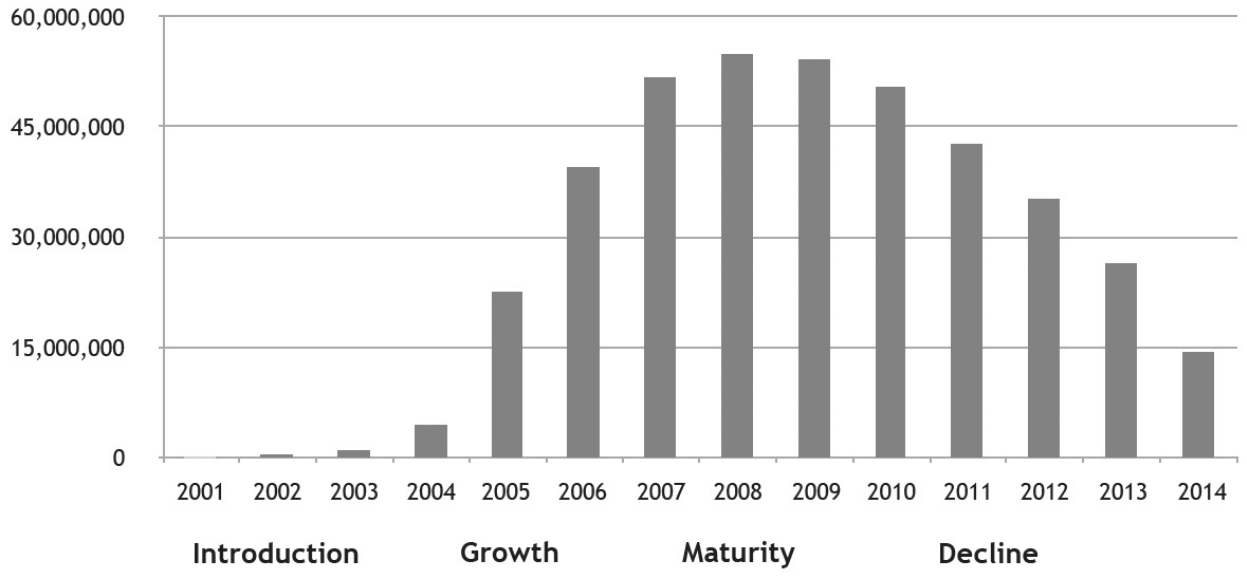


Figure 8: The Life Cycle of the iPod Family[24]

As Figure 8 shows, the iPod was launched in 2001 as Apple's first consumer music gadget. The company was a new entrant in the digital-music-player market, which at the time was dominated by products like the Nomad Jukebox from Creative Labs. In 2002, the iPod became Windows-compatible, and sales subsequently reached 600,000 units. In the following year, Apple launched iTunes, which helped sell more than 900,000 units in 2003 and nearly 4.5 million in 2004. The iPod had entered the growth stage. It had become the dominant digital-music player in the United States. To sustain growth, Apple enhanced the product and added new features, for instance, the ability to show photos and videos. The company also introduced new product variants, such as the iPod Nano and iPod shuffle in 2005 and the iPod Touch in 2007. Additionally, Apple issued several limited iPod editions, including a black-and-red U2 special edition. Sales of the iPod reached their peak in 2008, which also marked the product's maturity stage. In 2009, iPod sales started to decline. Consequently, Apple discontinued the original iPod, now called the iPod Classic, in 2014. As this example shows, even the most iconic products eventually decline and have to be replaced with new offerings. In Apple's case, this new product was the iPhone.

Let's now discuss the individual life cycle stages and explore how they influence strategic product decisions.

Development

Before the launch of a product, your primary goal is to find a product strategy that is likely to result in a product that is beneficial, feasible, economically viable, and ethical. In this period, you are likely to carry out product discovery and strategy validation work, and you may have to pivot—that is, to significantly change your strategy and choose a different path for attaining your vision. Take, for example, the idea mentioned earlier of creating a healthy-eating app. If it turns out that building an app is not a valid approach

—the market might be saturated with similar offerings, the technical challenges might be too big, or the app might be too difficult to monetise—I could pivot and choose to write a book on healthy eating instead.

While it is important that you spend the necessary time in the development stage and that you carry out the product strategy work required, you should avoid the mistake of trying to launch the perfect product. No product is impeccable from day one. Even a product like the iPhone had a comparatively humble start. Think of all the things the very first iPhone could not do: no videos, no copy and paste, and no third-party apps, to name just a few. The trick is therefore to launch a *good-enough* product aimed at the early market—the innovators and early adopters, and then to adapt and enhance it. Such a product is also referred to as the MVP, the minimum viable product.

How good your initial product, or MVP, has to be is closely linked with its innovation type. The initial version of a disruptive product can be comparatively basic, like the original iPhone. An adjacent product, however, faces higher customer expectations, as it addresses an established market where the customers have viable alternatives to choose from. Take the Google Chrome browser as an example. When the product was launched in 2008, it faced competition from several established products, including Internet Explorer, Firefox, Opera, and Safari. To succeed, Google had to offer a product that was clearly differentiated—a browser that was faster, more secure, and simpler to use than the competing offerings. The company also heavily advertised its product. I remember, for instance, seeing poster billboard ads of the browser at train stations in London.

A Short History of the Minimum Viable Product

The term *minimum viable product* (MVP) was originally defined by Eric Ries as "the version of a new product which allows a team to collect the maximum amount of validated learning about customers with the least effort." (Ries 2009c) This definition suggests that an MVP is a (throwaway) prototype created to test a specific strategic assumption. Consequently, people have suggested using, for example, videos, landing pages, and product fakes as MVPs. For example, a video might be used to test if there is sufficient interest for a new product, as Drobox did to test its original product idea. Over the following years, however, the commonly accepted definition changed, and people started viewing the minimum viable product as the initial product release—the product offered at launch.[25]

Introduction

After the launch, your objective is to achieve product-market fit and experience growth as quickly as possible. How long this is likely to take and how much effort it will require, depends on your product's innovation type. Building an initial customer base and finding out if and how people use the product is particularly important for disruptive innovations. Take Twitter as an example. The company had to discover how people employed the product to decide how to progress it, as Twitter's cofounder Ev Williams explains: "With Twitter, it wasn't clear what it was … Twitter actually changed from what we thought it was in the beginning, which we described as status updates and a social utility. The insight we eventually came to was [that] Twitter was really more of an information network than it is a social network. That led to all kinds of design decisions, such as the

inclusion of search and hashtags and the way retweets work."[26] Adjacent products, however, tend to require a shorter introduction stage, as they address an existing market. This allows you to learn more about the user and customer needs and how to best address them at the development stage.

With both disruptive and adjacent products, make sure you track the product performance and monitor how much value your product creates. If the curve is flat or if it rises only slowly, then you should investigate the causes and determine the right course of action. Sometimes, adapting the product is sufficient, for example, by enhancing and adding features. But other times, you require a more drastic change—a pivot. Flickr, for instance, changed from an online role-playing game to a photo-sharing website; YouTube evolved from a video-dating site to a video-sharing product. In both cases, a feature of the original product was unbundled and then released as a new standalone product thereby executing a pivot. In Flickr's case, this was the photo upload capability and in YouTube's, the ability to upload videos.

Another interesting example of a post-launch pivot is Google Glass, Google's wearable computer glasses. The headset started its life as a consumer product in 2013. But as the product attracted a significant amount of criticism and controversy—some Glass owners were insulted as "glassholes," and it was feared that the product could violate privacy laws—the company stopped offering the smart glasses in 2015 and moved them back into its research lab. Two years later, the product was relaunched as Google Glass Enterprise Edition, a B2B product aimed at people who need access to online information while using their hands to get a job done, for example, doctors attending to patients and assembly line workers who need to quickly look up information.

If pivoting is not an option or if you have already pivoted once or twice, then you should consider killing your product. While this might sound rather drastic, it frees up resources and avoids wasting time, money, and energy. Take, for example, Google Wave, a product that combined email, instant messaging, and wikis. Due to its lack of success, Wave was discontinued at the introduction stage about a year after its launch in 2009.[27] Remember that failure is part and parcel of the innovation process; there is no guarantee that your product will make it to the growth stage and become a success.

If you see a positive market response to your newly launched product, then that's great. But don't make the mistake of optimising it for the early market. The initial users and customers of a new tech product—who are also referred to as the innovators and early adopters—are usually happy to put up with a few teething issues as long as they will gain an advantage from using it. To get into the mainstream market, you have to satisfy much higher expectations. This includes providing a product that works flawlessly and is easy to obtain, install, and update.[28] Consequently, the transition to the growth stage may not be a small, incremental step. Instead, your product may face a gap or chasm between the early and the mainstream market, which you'll have to overcome (Moore 2006).[29] Figure 9 shows the product life cycle with a chasm between the introduction and the growth stage.

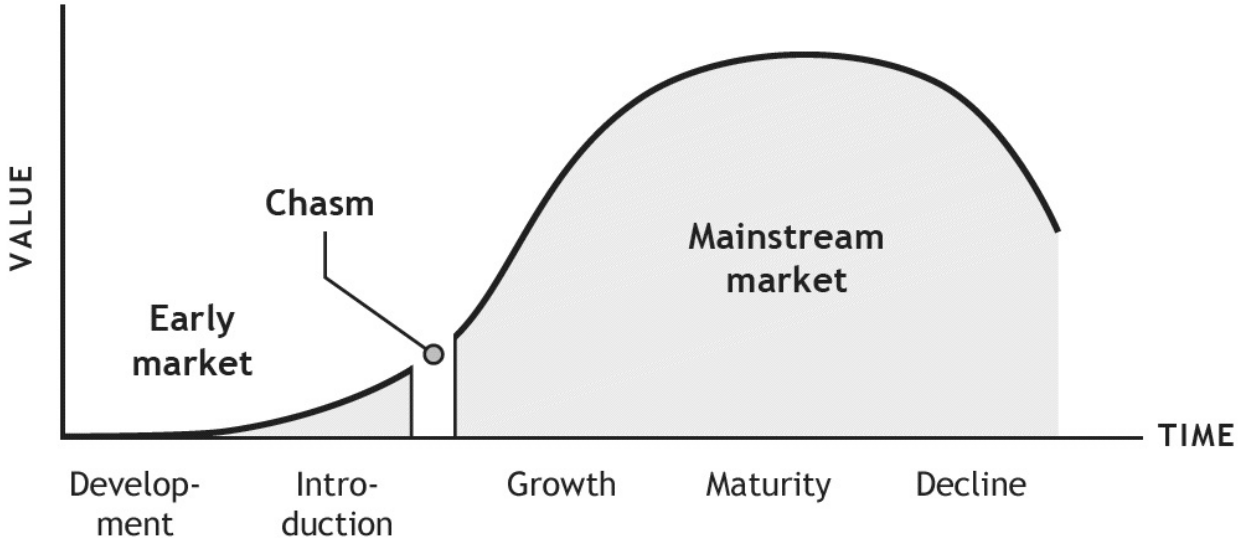


Figure 9: The Product Life Cycle Model with Chasm

To bridge the chasm, you have to adapt and improve your product. This may include enhancing the user experience, adding or improving features, refactoring the architecture or changing some of the underlying technologies, and adapting the business model.[30] The latter may include reducing the cost of acquiring customers and changing the marketing and sales channels. Take the Apple Watch as an example. The smartwatch was released in April 2015 as an iPhone accessory and lifestyle gadget. While the product was generally well received, Apple had to change its strategy and adapt the product in the coming years to achieve product-market fit. Initially, the company narrowed the target group and sharpened the value proposition of the smartwatch by focusing on the fitness market and turning it into a fitness device. This enabled the company to take the next step and add health features to the Apple Watch, thereby positioning it as a health and fitness product that served a larger market. At the time of writing, the latest version of the Apple Watch, series 7, works without an iPhone, measures heart rate and oxygen levels in the blood, comes with an ECG like feature, and offers a large choice of third-party apps. As this example shows, reaching product-market fit and entering the growth stage can be challenging, even for successful businesses and adjacent innovations. The product strategy is likely to be volatile in the introduction stage and may well require frequent updates to progress the product and enter the growth stage.

Growth

Once a product starts to experience significant growth, it has achieved product-market fit and entered the growth stage. To put it differently, the product now does a good job for the mainstream users and customers, and it offers the desired business benefits for the company. A revenue-generating product will have reached the break-even point by now and should produce a positive cash flow. The product strategy needs to focus on sustaining the growth, penetrating the market, and fending off competitors. You will therefore have to find ways to attract more users and customers and to keep your product

clearly differentiated. Chances are that the competitors will start to copy some of its features. Think of what happened in the smartphone space: The design of the original iPhone that was once a major differentiator has become the standard for all modern phones. Or take Samsung's innovative three camera smartphones, which quickly faced competition from iPhone models also equipped with three cameras.

While you typically want your product to enter the growth stage as quickly as possible, this does not come without challenges: You now have to deal with a product that serves a larger, growing audience with diverse needs, that is becoming increasingly feature-rich, and that requires more people to develop it. You should therefore consider the following two techniques to sustain the growth: First, you might unbundle your product, spin off one or more features, and turn them into a new product—as Facebook did with Messenger. Second, you could create product variants and develop specialised versions of your product for a specific market segment, think of YouTube Kids, for example. I'll explain both techniques in more detail later in this book.

Additionally, you should carefully determine how many product people you need to manage the growing product and guide the various development teams and how the individuals should work together. I find it helpful to have one person in charge of the overall product. The individual works with additional product people who own product parts, like features or components. If you use Scrum, applying this model may result in having an overall Scrum product owner who collaborates with several feature and component owners (Pichler 2016). I find that this setup is well suited for the growth stage where the product strategy still changes—although not as much as in the development and introduction stages. Employing a strategic and tactical product role, as suggested by the scaling framework SAFe, is more appropriate for mature products in my experience where the product strategy is stable, assuming that you do not opt for a life cycle extension.[31] No matter which scaling approach you choose, don't wait until you have reached product-market fit before you decide how to share the product management work and look for the right product people to work with you. Start this process in the introduction stage when you notice a sustained rise of your product's value curve.

Maturity

Despite your best efforts, growth will eventually start to stagnate, and your product will enter maturity. When this happens, you face an important strategic inflection point with two options: extend the product life cycle or turn your product into a cash cow. Let's look at these options in more detail.

Option 1: Life cycle extension

Your first option is to move the product back into the growth stage thereby extending its life cycle, as Figure 10 shows.[32]

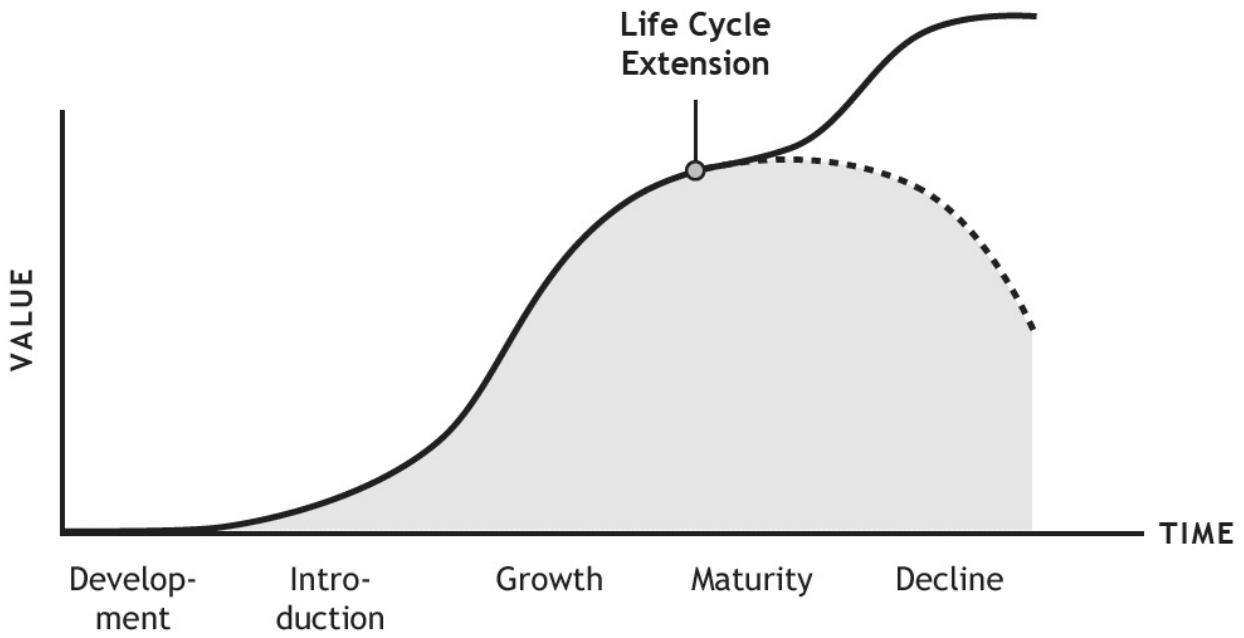


Figure 10: Extending the Product Life Cycle

Several techniques can help you make an ageing product attractive again. Let's look at four of them. First, you might choose to enhance its capabilities and add new features like Apple has done with the iPhone. The company introduced apps, increased the product's size, improved the camera, and added face recognition, to name just a few. Sometimes, though, the opposite strategy is more appropriate. Instead of adding more features, you may want to remove some and declutter your product. Take Microsoft Word, for example. Microsoft has made significant efforts to simplify the application in recent years, thereby making it easier for people to use the product. A third way to stimulate growth is to take your product to a new market or market segment. Think of YouTube Premium, a variant of YouTube, for instance. At the time of writing, the product allows people to watch YouTube videos without ads and to download them, as well as listening to and downloading songs on YouTube Music. Finally, you might consider bundling your product with other offerings to increase its attractiveness. For example, iOS and Android bundle a mobile operating system with several pre-installed apps including a web browser, email client, and maps.

While rejuvenating the product and moving it back into the growth stage may sound like a good idea, it is not necessarily the right thing to do for your product. If you should choose this option depends on three main factors:

1. The category the product belongs to continues to be attractive.
2. You don't need to turn the product into a cash cow.
3. You can invest the time and money required to extend the life cycle.

If the category your product belongs to has lost its attractiveness, then revitalising the product will be difficult. Take MP3 players, for instance. The category has lost its appeal.

Most people listen to music on their phones instead of using dedicated MP3 players. If, say, Apple wanted to revitalise its iPods, then this would be hard to achieve (unless the company repositioned them as niche products targeting audiophiles).

Additionally, a life cycle extension only makes sense when you don't need to turn your product into a *cash cow*. As its name suggests, a cash cow is a product that offers plenty of business benefits and requires a comparatively low investment. Any product portfolio should contain a healthy mix of younger and older products (Henderson 1970): Older products essentially act as core innovations and pay for the development of new ones. It can therefore be advantageous to accept maturity and let your product age, as I discuss below.

Finally, extending the product life cycle can require a substantial amount of time and money: You may have to carry out market, user, and competitor research; you may have to offer new features; you may have to evaluate new technologies and integrate them into the product; and you may have to adapt the business model. What's more, the longer your product stays in maturity and the poorer the code quality is, the bigger this effort is likely to be: Chances are that the market has moved on while you've focused on smaller, incremental enhancements to secure your product's current position. Shifting gears, embracing an entrepreneurial mindset, and innovating the product can then be challenging. Similarly, a product with a poor software quality—for instance, with plenty of bugs and spaghetti code—makes a life cycle extension harder and more expensive. You may have to carry out architecture refactorings before you can add new features and enhance existing ones.

Option 2: Cash cow

Your second option is to let your product mature and use it as a *cash cow*. Such a product creates plenty of business benefits, but it requires only a moderate to low investment. This option is advisable when a life cycle extension is not required. This means that you have other products in the pipeline, which will eventually replace your product, or that you are able to secure new products through an acquisition.

When you use a product as a cash cow, it becomes a core innovation. Consequently, you should adopt a more conservative mindset and play a defensive game: You typically want to protect your product's position in the marketplace while minimising the amount of money you spend on it. This often results in incremental enhancements and bug fixes rather than bigger changes like adding brand-new features. It might also make it attractive to outsource some of the development work if this helps decrease cost. But to be able to milk a product for an extended period and prevent early decline, you must not become complacent and neglect it. While the product strategy is likely to be stable at this stage, you should keep an eye on market developments and monitor your competitors; continue to track the product performance and regularly check if the current strategy is still working; and invest the money required to keep the product beneficial for its users.

Decline

Even if a mature product is well looked after, one day it will reach the decline stage—the value it creates starts to decrease. Think of the iPod Classic, Apple's original MP3 player. The product dominated its category at one stage, inspired many product designers, and acted as a massive revenue source for its company. But after years of declining sales, Apple finally retired the iPod Classic in 2014.

Once your product has entered the decline stage, you have, again, two main choices: Your first option is to accept the decline as the final stage in your product's life. If that's your choice, consider what retirement means for the product. Some B2B products, for example, in the telco and healthcare space, are sold together with service contracts. These usually require maintaining and servicing the products after the sale has ended. In these cases, *end of sales* and *end of life* will differ, and you will have to plan how you can retire the product with minimum investment and at the same time, honour the existing agreements. This might include outsourcing or offshoring (some of) the maintenance work. Additionally, explore how you can encourage existing customers to upgrade to a new product, for example, by offering a special discount, assuming that such a product will be available.

Alternatively, you might want to attempt "resurrecting" the asset by re-positioning it as a niche product. Take turntables, for example. When I was a child in the 1970ies, turntables were ubiquitous. I had a small one, too, and I would play my favourite records for hours. Then came the cassette tape, the CD player, and the music streaming services, and the sales of turntables dwindled until the product category seemed all but extinct. But while the new technologies were making it increasingly convenient to listen to music, turntables started to reappear—initially, as a niche product aimed at audiophiles who longed for the sound quality records have to offer. Surprisingly, though, turntables have staged a remarkable comeback and record sales have grown to 27.5 million in 2020 the US alone.[33] As this example shows, re-positioning a declining asset as a niche product can give a declining product a second lease of life.

Summary

Table 2 summarises how the life cycle stages shape the product strategy.

Table 2: The Product Life Cycle and the Product Strategy

Life Cycle Stage	**Strategy**
Development	Create a validated strategy: a strategy that is likely to result in a beneficial, feasible, economically viable, and ethical product.
Introduction	Adapt and improve your product to achieve product-market fit (PMF). This may require incremental changes such as enhancing the user experience, adding new features, refactoring the architecture, and adapting the business model to ensure it is scalable. But it may also require a more drastic change, a pivot. Aim to achieve the break-even point for a revenue-generating product by the end of this stage. Decide how to share the increasing product management work, assuming that it will be too much for one person once the product has entered the growth stage.
Growth	Sustain the growth by penetrating the market and fending off competitors. Keep your product attractive and continue to progress it. Add new features and enhance existing ones; improve the user

	experience. Consider unbundling your product and creating variants to manage the growth. If your product generates revenue, ensure it is profitable.
Maturity	As growth stagnates, extend the life cycle and revive growth by taking the product to a new market, for example, or bundling it with another product or service. Alternatively, milk your product and turn it into a cash cow. Defend its market share and focus on maximising profitability for revenue-generating products.
Decline	Reduce cost to keep the product economically viable for as long as possible, then start phasing it out. Entice existing customers to upgrade to a new offering. Consider outsourcing or offshoring some development work. Alternatively, "resurrect" your product and reposition it as a nice offering.

While the product strategy should become progressively less volatile as your product grows and eventually matures, it will continue to change and evolve, as table 2 shows. You should therefore regularly assess your product's performance and adjust your strategy accordingly—no matter in which life cycle stage it is. Strategy and execution go hand in hand for digital products, they are two sides of the same coin. The strategy work isn't over until the product is retired.

Collaborate with the Stakeholders and Development Teams

The most amazing product strategy is useless if the stakeholders and development team members don't support and follow it. Securing their buy-in and creating a shared strategy is therefore truly paramount. The following paragraphs provide techniques to help you with this.

Stakeholder Identification

A stakeholder is a person who has a stake in your product—anyone who is affected by it or shows an interest in it. While this definition includes users and customers as well as development teams, I use the term to refer to the business stakeholders. For a commercial product, these are likely to include representatives from marketing, sales, and support. But they might also include employees from legal, finance, and human resources. For an in-house product, your stakeholders may be people from the affected business units, operations, and line management.

Stakeholder Analysis and Engagement

Once you have identified the stakeholders, you should determine how to best engage them. You can do this by using the Power-Interest Grid, described in Eden and Ackermann (2011). The grid analyses the stakeholders by considering their power and their interest; it assumes that stakeholders take a low or high interest in your product and that they have low or high power. This results in four stakeholder groups: players, subjects, context setters, and the crowd, as Figure 11 shows.

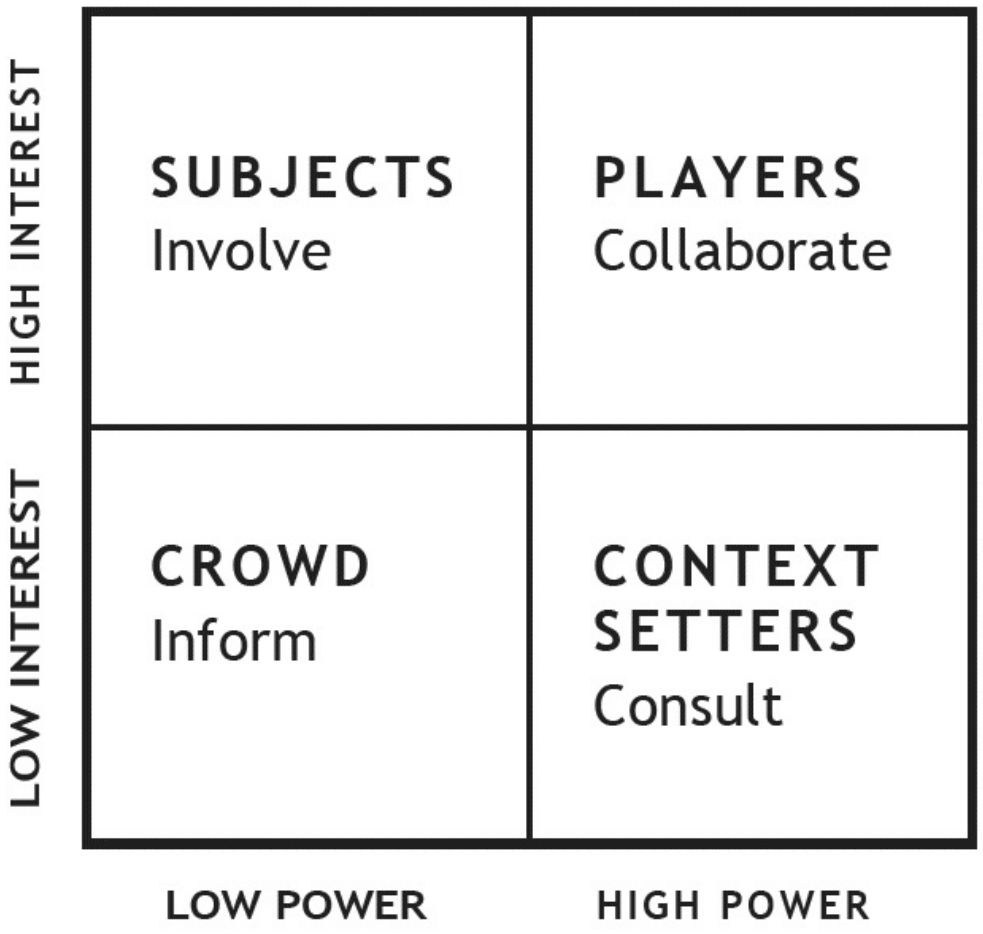


Figure 11: The Power-Interest Grid

Stakeholders with high interest and high power are called *players*. These individuals are the key stakeholders, and they are important partners for you: They should help you create and update the product strategy and product roadmap. Aim to secure their buy-in, leverage their ideas and knowledge, and establish close and trustful connections with them. Additionally, try to keep the group of players stable so that the individuals work with you on a continuous basis. This avoids loss of knowledge caused by handoffs, it makes it easier to establish trust, and it facilitates effective collaboration and decision-making. It would be undesirable, for instance, to have the marketing group send a new representative every time a product strategy workshop takes place.

Be aware, though, that collaboration requires leadership. As the person in charge of the product, you should be collaborative and cultivate an open mind. Aim to build consensus with the players, but don't shy away from difficult conversations. Don't settle for the smallest common denominator. Have the courage to decide if no agreement can be achieved. Great products are not built on weak compromises. As the saying goes, "A camel is a horse designed by committee."

Subjects are individuals with high interest but low power, for example, users of your product and people who work on related products. These individuals feel affected by the

product and are keen to influence it. But they are not able to affect decisions. Subjects can offer valuable feedback, though, and they can help you secure understanding and buy-in for your product across the business. Keep them involved, for example, by inviting them to participate in a user forum, early adopter program, and/or selected sprint review meetings.

People with low interest but high power are called *context setters*. They affect the product's context, but they usually take little interest in the product itself. Context setters are often powerful senior and executive managers who might make your life difficult if they are not on your side. Regularly consult them to ensure that their opinions are heard, for example, by inviting them to sprint reviews or having one-on-one meetings. But don't let the context setters intimidate you and don't allow them to force product decisions on you.

Everyone else is part of the *crowd*. As these individuals are not particularly interested in your product and don't have the power to influence product decisions, it's usually sufficient to keep them informed, for instance, by sharing a quarterly update with them.

Two Common Stakeholder Management Pitfalls You Should Avoid

There are two common stakeholder management mistakes I see product people make: The first one is to please the stakeholders, say yes to all ideas and requests, and broker weak compromises. This might create a product strategy that makes the stakeholders happy. But it's unlikely to achieve sustained product success. The second mistake is to create the product strategy without stakeholder involvement, present them with a finished plan, and expect the individuals to agree with it. This fails to leverage the knowledge and expertise of the stakeholders. What's more, it makes it unlikely that the stakeholders will fully support and implement the product decisions, as they haven't had the opportunity to share their views and concerns.

To avoid these pitfalls, take a genuine interest in the ideas, concerns, and needs of your key stakeholders; build trustful connections with them; and involve them early and frequently in the strategizing work. But if you are faced with pushy individuals who insist that their requests must be implemented, then do not simply give in. Don't appease them and don't avoid a difficult conversation. Instead, patiently listen to them and thank them for sharing their ideas. But don't be afraid to say no, as long as you clearly explain why you are declining a request and have data to back up your arguments. Remember that your job is to make the product successful, not to please the stakeholders.

Collaborative Workshops

Running a collaborative workshop is a great way to create a product strategy, be it for a new product or a significantly changed one. Its objective is to establish an initial, testable, and agreed strategy. If you are working on a brand-new product, then you can also use the workshop to come up with a shared and inspiring product vision.

Running such a workshop, be it online or onsite, offers three key benefits: First, it helps you make better decisions. This is achieved by leveraging the collective creativity and expertise of the attendees. Second, it creates a shared understanding and aligns people. Everyone understands the resulting product strategy and knows the decisions it is based on. Third, it results in stronger buy-in: Inviting people to contribute to a decision usually generates more support and it increases the chances that people will follow the strategy, rather than paying lip service to it.

To experience these benefits, apply the following seven tips—which are based on my book *How to Lead in Product Management*.

1. *Involve the right people*: These are the key stakeholders, development team representatives, and yourself. You may also want to invite subjects and context setters if their contribution is required. But avoid workshops where people are present who are not needed to make the right decisions. If you find that you can hardly make a product decision without a small crowd being present, then explore why this is. For example, the individuals might not trust you to consider their concerns and needs, or you might not be fully empowered to make the necessary decisions.
2. *Employ a dedicated facilitator*: Ask your Scrum Master or another qualified facilitator to help prepare and facilitate the workshop. This includes setting ground rules, helping you choose an effective decision rule (see below), guiding the workshop attendees through the decision-making process, and ensuring that everyone is heard, and nobody dominates. This makes it more likely to have a successful workshop, and it allows you to focus on contributing to the product decisions rather than having to facilitate at the same time.
3. *Forster a collaborative mindset*: Encourage the workshop attendees to actively contribute to the product strategy. Listen attentively to what the individuals have to say. Keep an open mind. Avoid clinging to preconceived notions and ideas and be willing to change your mind when appropriate. Show appreciation for people's suggestions even if you disagree or if you don't find them helpful. This will make the individuals feel understood and show them that you value their contributions.
4. *Select a decision rule*: Such a rule clearly states who decides and how you can tell that the decision has been made. Helpful decision rules for a product strategy workshop are *unanimity*, *consent*, and *product person decides after discussion*. The first rule requires everyone to be happy with a decision; the second one means that nobody has any meaningful objections; the third rule, finally, asks you to decide after you have listened to the ideas and concerns of all attendees. No matter which rule you use, the goal is to make the right decisions—decisions that maximise the chances that the product will create the desired value and at the same time attract as much support from the stakeholders and dev team members as possible.
5. *Base decisions on empirical evidence*—rather than gut feeling or the highest paid persons' opinion (HiPPO). If you lack the relevant data to create an initial product strategy, then carry out just-enough upfront discovery and market research work before you continue the strategy creation work.
6. *Don't rush important decisions*: Take the time required to gather everyone's perspectives and build a shared understanding before you try to reach agreement —even if you will make the decision as the person in charge of the product. This will help you make the right decision and secure as much buy-in as possible.
7. *Use a helpful tool*: Use a tool like the product vision board, which I'll discuss in the next chapter, to structure the conversation and capture your ideas. If you are

unsure which tool to use, pick one that resonates with you and try it out. Ensure that everyone attending the workshop has access to the tool and understands how to use it. This is particularly important when the workshop takes place online.

Tips for Encouraging the Right People to Attend Strategy Workshops

Sometimes stakeholders and development teams can be hesitant to participate in strategy workshops. If that's the case, find out why that is. Maybe the individual does not fully understand why she or he is required; maybe the person had bad experiences in the past contributing to important decisions; maybe the individual is pressed for time; or maybe there is a conflict with another attendee. Once you understand the reason, ask how you can make it easier for the individual to attend the workshops, for example, by timeboxing the meetings.

If the opposite is the case, and you find that you can hardly make a strategic product decision without a small crowd being present, then there are likely to be different causes at play. Maybe some of the individuals do not trust you to consider their concerns and needs; or maybe you are not fully empowered to make the necessary decisions. Once you've identified the root cause, consider how you can best resolve it.[34]

PRODUCT STRATEGY DEVELOPMENT

Doing the right thing is more important than doing the thing right.
Peter Drucker

In this chapter, I discuss a range of product strategy practices to help you make the right strategic decisions, such as finding the right audience for your product, discovering a need that's worthwhile to address, and making sure your product stands out from the crowd. Most of the practices are applicable to the entire product life cycle, from brand-new products to mature ones. Let's dive into them.

Segment the Market

Segmenting the market means dividing the potential users and customers into distinct subgroups. This helps you to create a focused product with a compelling value proposition and a great user experience. Your segments should be clear-cut so they do not overlap. To put it differently, you should be able to tell who belongs to a segment and who does not. What's more, each segment should be homogenous, and the people within it should respond to your product in the same way. Take my healthy-eating app mentioned earlier. A large group of people could benefit from it, including teenagers who suffer from an eating disorder and adults who have type-2 diabetes. Trying to please everyone with the same product would not only be challenging; it would also result in a feature-rich offering that might not do a good job for anybody.[35]

Segmenting by Customer Properties and Benefits

How you segment the market matters. The segments not only define who the users and customers are, but they also influence many product decisions. While there are different ways to divide up the market, you face two basic choices. You can form segments that are based on the customer properties, or on the benefits that your product provides. Common customer properties include the following three items:

- Demographics such as age, gender, marital status, occupation, education, and income.
- Psychographics, including people's activities, interests, and opinions.
- Geographic regions like Europe, Middle East and Africa (EMEA), and Asia-Pacific (APAC).

For business-to-business (B2B) products, there are two additional qualities:

- Industries or verticals, for instance, automotive, education, and health care.
- Company size, such as small and medium-size enterprises (SMEs).

While the attributes listed above differ, they have something important in common—they all concentrate on the customer, be it a consumer or a business. Let's use my healthy-eating app again. To segment the market, I could choose demographic and psychographic attributes and define my target group as men aged 20–30 who are single, work long hours, don't exercise much, and eat out frequently.

An alternative approach is to divide the market using the benefit the product provides or the problem it addresses (Christensen and Raynor 2013). This suggests that you first and foremost consider people's needs. For example, if the main benefit of my healthy-eating app is to help people improve their eating habits, then I might identify the following three groups: teenagers who suffer from an eating disorder, single men aged 20–30 who have poor eating habits and want to lose weight, and middle-aged adults who have type-2 diabetes. Following this method is likely to result in different products or product variants.

The great thing about benefit-based segmentation is that it reduces the risk of overlooking people who are likely to take advantage of your product; it also offers you the opportunity to reconstruct the market boundaries. Take the Nintendo Wii game console, for instance, which was released in November 2006. Rather than using customer properties such as hard-core and casual gamers and trying to compete in these segments against Sony PlayStation and Microsoft Xbox, Nintendo looked to the gaming industry's noncustomers and investigated what prevented people from playing video games. This allowed the company to redefine the market boundaries and to connect with older people and young children: two groups that share few demographic and psychographic attributes. The result was an innovative console that people could interact with in novel ways—through motion control and a magic wand rather than a keyboard or a specialised controller.

Whichever way you segment the market, avoid the following two mistakes: First, don't blindly follow predefined segments. I have seen product managers cling to existing customer-based groups while trying to create new, innovative products. Unsurprisingly, the outcomes were poor. Second, don't discard an idea because it does not fit into a set of predefined segments. You may miss opportunities to create a new product or to discover new markets—as Apple did with the iPhone and Nintendo with the Wii.

Choosing the Right Segment

Segmenting the market often results in several groups that your product could serve. In the case of my healthy-eating app, these might include teenagers with an eating disorder, single men aged 20–30 who want to lose weight, and middle-aged adults who are at risk of developing type-2 diabetes. Which one should I pick? To select the right segment, evaluate the different groups and opt for the most promising one. A great tool to do this is the *GE/McKinsey matrix* (Coyne 2008). While the tool was originally developed to assess a business portfolio, it can also be applied to market segments. It encourages you to assess

how attractive the segments are and how well your business can serve them, as Figure 12 shows.

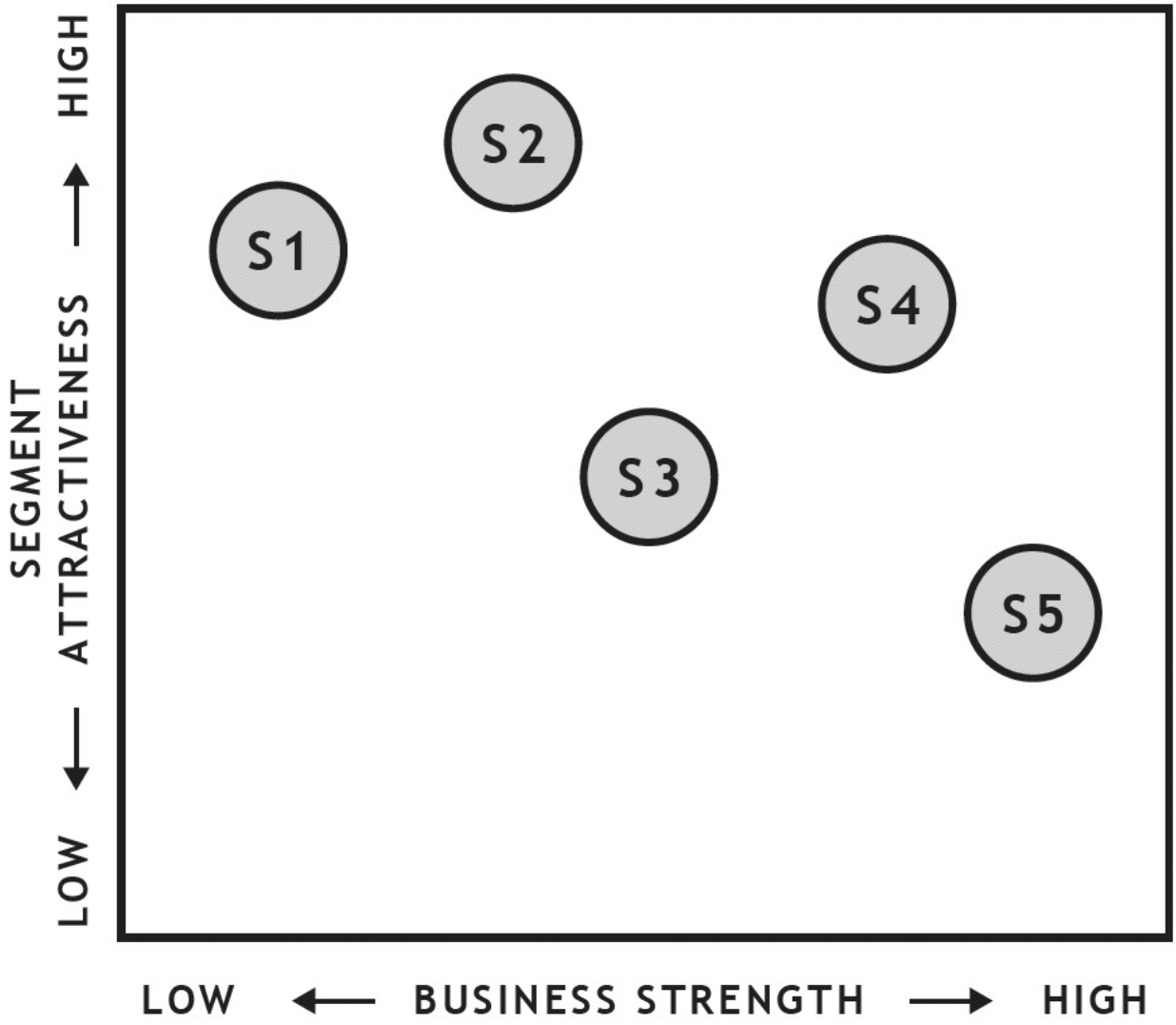


Figure 12: The GE/McKinsey Matrix

The GE/McKinsey matrix in Figure 12 states the attractiveness of the market segment on the vertical axis, and the strength of the business on the horizontal one. It shows five sample segments that are ranked according to the two dimensions. The most promising one is S4: it is attractive, and the business can serve it. S1 and S2 are more attractive than S4, but the business' ability to serve them is significantly lower. The remaining two segments are less attractive.

To determine the attractiveness of a segment, consider the following four criteria:

- *Need*: How strong is the need? How much will the individuals benefit from having it addressed?

- *Size*: How large is the segment, how many people does it contain?
- *Growth*: Does it show signs of growth or expansion?
- *Competitors*: Who are the main competitors, and how fierce is the competition?

To understand the business strength, investigate your ability to serve the segment. Answering the following questions can help you with this.

- *Customer relationships*: Do you have the necessary expertise to serve the segment? If not, can you obtain the knowledge and skills required? Do you have the right marketing and sales channels in place to reach the right people? If not, what will it take to establish them? How difficult and expensive is it to acquire customers?
- *Entry barrier*: Are there any barriers for entering the segment including regulatory requirements—assuming that you haven't served it?

Combining the GE/McKinsey matrix with an iterative strategy-validation approach—which I'll explain in more detail later—means that a qualitative, informal evaluation of the segments is usually sufficient. Here is how this combined approach can work: Choose the most promising segment and then carry out the necessary validation work. Test, for instance, that the segment is sufficiently large, that the need is strong enough, and that you have the people, technologies, and channels to create and deliver the product. If the validation is unsuccessful, then select and test the next segment. Say that I decide to first address middle-aged adults who have type-2 diabetes with my new healthy-eating app. I would then validate this segment and carry out some direct observation, problem interviews, and competitive analysis, for example. If the validation failed, I would test the next segment, for example, teenagers suffering from an eating disorder. I would continue this process until I've found an attractive segment—or decide to pivot and fundamentally change the product strategy.

Find an Itch That's Worth Scratching

Discovering a problem that people want to see addressed or a benefit that they would no longer want to miss once they've experienced it is a crucial step to achieve product success. Take the example of Sonos, a wireless hi-fi system that allows people to enjoy music by providing easy access to a range of streaming services. It's simple and convenient to use, and it offers a decent sound quality. Products like the Sonos system are sometimes called *vitamins*, as they provide a nice-to-have benefit, similar to vitamin supplements. Contrast this with a search engine like Microsoft Bing or Google Search, which solves the problem of finding information on the Internet. Such a product is also referred to as a *painkiller*, as it addresses an issue or pain point. But no matter how a product is classified, it must create tangible value for its users—or it will fail. A vitamin must be "sticky" enough so that people come back to it and keep using it; a painkiller must do a good job at addressing a problem people want to have solved once they've become aware of it.

Establishing the Right Mindset

The most powerful techniques to discover and describe user and customer needs are ineffective if you haven't established the right attitude. The following four guidelines will help you cultivate the right mindset:

1. *Let go of any product ideas and business goals for now*. If you are driven by your desire to make a specific idea work, you will find it hard to empathise with the target users and customers. You are likely to lack the necessary receptivity and open-mindedness, and you are in danger of suffering from confirmation bias and discard data that challenges your ideas.
2. *Empathise:* Develop a genuine interest in the people your product might serve and aim to understand their feelings, needs, and interests. Explore how they currently solve a problem or achieve a benefit. Reach out with humbleness and the intention to learn. Being empathic means to care about the other person—no matter if we agree with their views or if we like or dislike them.
3. *Keep an open mind:* See the discovery of user needs as a learning process and be willing to experience failure to acquire new knowledge. For instance, your initial need or target group might be wrong. But that's OK as long as you are willing to challenge your preconceived ideas and assumptions and change your mind based on the data you have collected.
4. *Take the time required and don't rush the work*. If you feel pressured to quickly come up with results or if you have other duties that require your attention and distract you, then it will be difficult to develop the necessary receptivity and curiosity. But don't procrastinate and don't be a perfectionist. Your goal should be to come up with an initial, good-enough needs statement that can be validated and subsequently corrected and refined.

Discovering a Need

How you approach the discovery of the right need will depend on your starting point. If you have a specific need in mind, for instance, to "help people reduce the risk of developing type-2 diabetes," then look for the group who would benefit most from having the need addressed and who you can serve, as I explained in the previous section. If you start with a specific product idea like a healthy-eating app or a business goal such as "creating a new revenue source," ask yourself who might benefit from such a product or who you can serve to generate the desired value for the business. Finally, if you have selected a target group, for example, "adults aged 40 to 55 years with unhealthy eating habits," then explore how you can add value for the individuals and which specific needs you could address. Regardless of you approach to discovering the right need, you will benefit from using the following three techniques:

1. *Directly observe prospective users and customers and talk to them* to find out if the need does exist, if people are aware of it, and to which extent and how it is being served at present. What feelings are they experiencing when they use a

product to get a job done? How happy or discontented are they? This will allow you to empathise with the individuals and understand if they would benefit from having the need addressed, possibly in a better, cheaper, or more convenient way. Be careful, though, not to discuss any ideas about your product or its features. Your objective at this stage is not to validate the solution but to discover a need.[36]

2. *Talk to the support team and read customer-complaint emails* to find out where users and customers experience friction and what they commonly complain about. What do people often struggle with? Which recurring problems do they encounter? What frustrates them? Additionally, use analytics data—if available—to explore common user journeys and discover frequent errors and drop-offs.
3. *Build on your own experience*, assuming it is relevant. For example, if I want to help people eat more healthily, I might reflect on my own eating habits. I might explore what has helped me in the past to improve them and eat more healthily. Be careful, though, not be swayed by your own preferences and biases, which can distort your ability to clearly see the needs of others.

In the context of the overall approach suggested in this book, your aim is to come up with an initial needs statement that is good enough so that you can validate it—that is, you can show that the need exists for the target group selected and that you can successfully address it. The validation work may require changing and refining the needs statement as well as adjusting the market segment or choosing a different one. In other words, discovering the right need is often not a linear process but an iterative one that involves an element of trial-and-error, as I explain in more detail in the chapter *Product Strategy* Validation.

Using Empathy and Consumption Chain Maps

Two useful tools to capture how users and customers currently get a job done and to determine which needs should be addressed are *empathy* and *consumption chain maps*. Additionally, the tools can guide your research work and remind you to take a holistic look at how people get a job done or interact with a product.

Empathy map

An empathy map is a simple visual that captures your observations about one or more users or customers (Gray 2017). As its name suggests, the map wants to help everyone involved in the development effort empathise with the individuals. In other words, it wants to help the stakeholders and dev team members understand the users' and customers' feelings and needs. While empathy maps can vary in form and content, I like to work with the one shown in Figure 13.

The empathy map in Figure 13 contains five pieces of information. It states who the user or the customer is, which may include relevant demographic information in addition to the individual's name; it captures what the individual says and does; it describes the person's feelings; and it articulates the problem they want to get solved or the benefit they want to obtain.

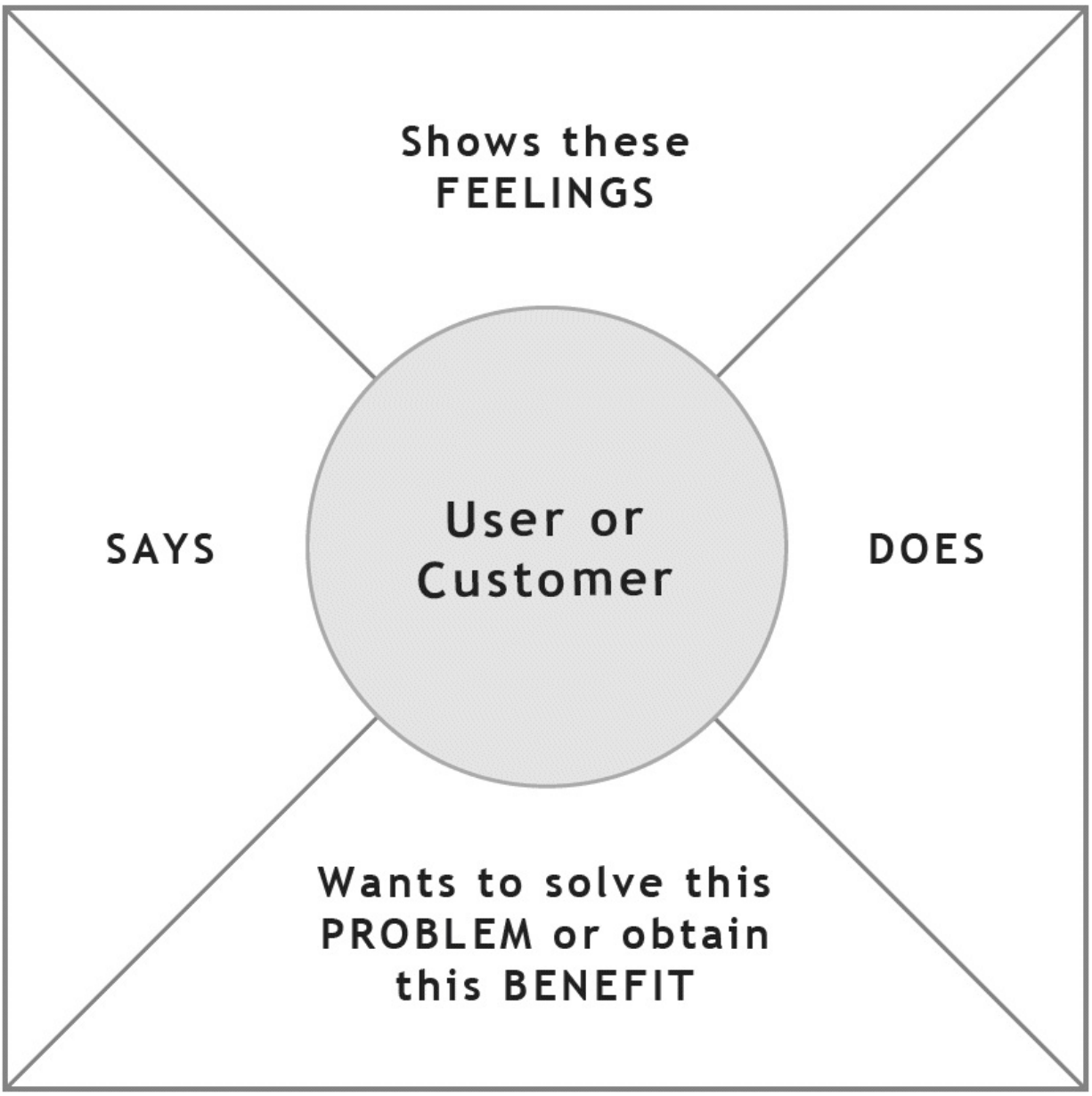


Figure 13: Empathy Map

When creating an empathy map, only capture information that you have *directly observed*. For example, pay attention to voice pitch and volume, facial expressions, gestures, stance, and other body language elements to discover the feelings of the user or customer. Don't make the mistake to speculate and to project feelings onto the individual, for example, believing that someone who sighs while using a product must be frustrated.[37] If you are in doubt, ask a clarifying question. You might say, for example, "I noticed that you sighed. Is there a specific reason for it?" Additionally, consider combining maps that show similarities into a more generic one. This will help you identify common behaviours and needs, and it will help you create personas—should you choose to do so.

Consumption chain map

A consumption chain map visualises how users currently interact with a product (MacMillan and McGrath 1997). The idea is to map the entire experience with a product —be it your product or a competing offering—and to use the visualisation to identify better ways to address a need.[38] To create the map, carry out direct observation and possibly shadow prospective users and customers. Then visualise the touch points they have with the product and link them together in a chain, as Figure 14 illustrates. Note that the map is generic. When you use the tool, describe the specific interactions people have with the product, for instance, users might evaluate the product by watching a YouTube video or reading a review on a website.

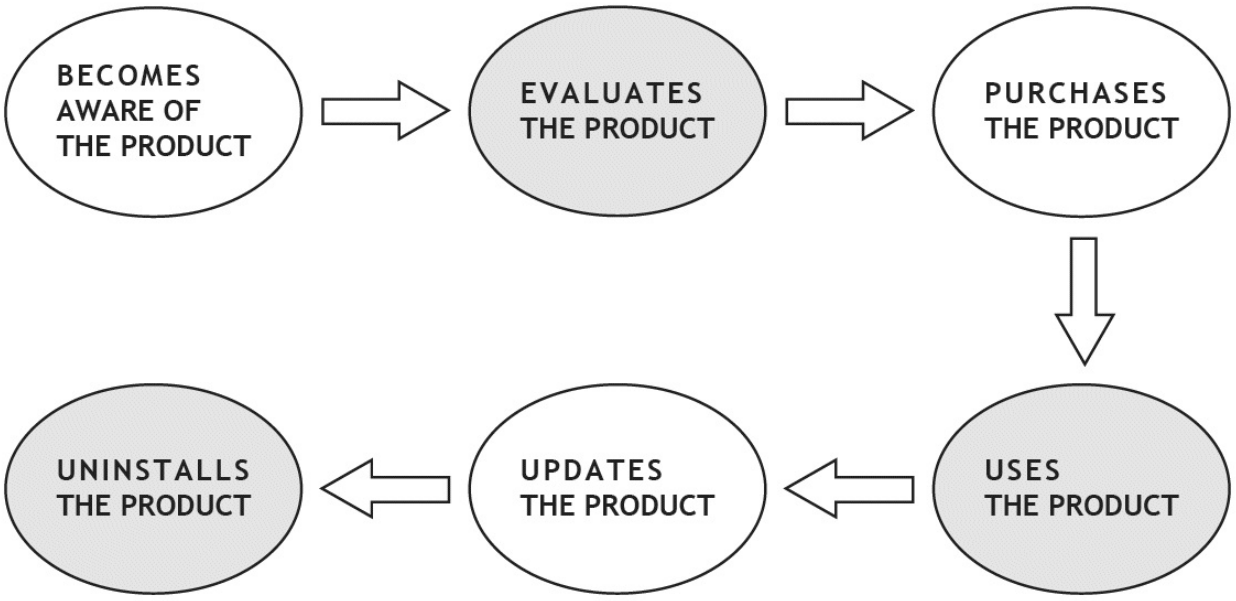


Figure 14: Consumption Chain Map

A consumption chain map—like the one in Figure 14—typically starts with a description of how people become aware of the product, for example, through the recommendation of a friend or seeing an ad on a social media platform, and it ends with the user no longer using the product and uninstalling it in the case a native app. To describe the entire consumption chain, you might have to look beyond a single product and consider how different products and possibly services interact. Take Microsoft Office as an example. To describe its consumption chain, we may have to start with prospective customers carrying out an Internet search, visiting the Office website, and considering the options available to them before agreeing to subscribe to the suite and download it. In this example, we would have to consider three assets: a search engine, a website, and a product bundle.

Once you've created a consumption map, analyse the customer experience at each link and ask yourself the following questions: How much time does it take on average to carry out the steps? Which steps add value from a user/customer perspective? Where do

problems occur? Where do users get confused, impatient, or frustrated? Then add the information to the map.

Next, identify how you can do a better job at meeting the user and customer needs and improve the customer experience. Explore how you can provide what the customers want—where and when they want it—without wasting people's time or making life difficult for them (Womack and Jones 2005). For instance, can you remove steps that are not value creating? Can you eliminate any waiting and delays and reduce the time needed to get the job done? Can you prevent errors and problems from occurring? Or can you address an existing need in a cheaper way? Be careful, though, not to become solution-focused at this stage and don't start designing the product. Rather pay attention to the need you want to address.

Making the Need Specific

Once you have found a need that your product should address, describe it clearly and specifically. Let's say that I am following the vision of helping people eat more healthily. The need I have identified is "help me improve my eating habits." This might be a nice, initial statement, but it is too big and vague. It does not say which eating habits should be improved and what the expected benefits are. To make the description more specific, I might refine the original statement and say, "help me reduce the risk of developing health problems related to my diet." This statement is better; it is clearer, and it mentions a desired outcome or benefit. But it is still not specific enough. I therefore have to refine it further, and I might change it to "help me reduce the risk of developing type-2 diabetes." This sounds like a good enough need description to me—a statement I can now validate. I could, however, try and make it even more precise by stating a target. This might result in "help me reduce the risk of developing type-2 diabetes by 20%." As the example shows, it is not uncommon to start with a vague and big needs statement and to iteratively refine it until it is specific enough. Applying the following three tips will help you with this.

1. *Describe what success looks like from the users' and customers' perspective*. To do so, consider using the template *help me <desired benefit or problem to be solved>*, which I applied in some of the examples above.
2. *Make the statement testable* so that you can show that the need is valid and that you can identify any risks that they may contain, as I describe in more detail in the chapter *Product Strategy Validation*.
3. *Capture an outcome, not an output or a feature*. Your needs statements should answer the question of why people would want to use and pay for the product, not what the product should do and which functionality it should offer. For instance, being able to see and analyse eating trends and to sync data with a smartwatch are features in my mind, not needs.

If you struggle to describe the need so that they are clear and specific, then you may have to carry out more discovery work and, for example, spend more time observing and interviewing users and customers before you can continue to refine the statement.

Selecting a Primary Need

If you have found more than one need that you want to address—which is not uncommon, then determine the *primary* need. This is the main reason for people to use or buy the product. If this need is not addressed, then the members of the target group will have limited motivation to engage with it or purchase it.[39] Identifying the primary need helps you create a compelling value proposition; it facilitates designing the right user experience; and it makes it easier to come up with the right marketing messages. Say I have discovered that some target users want to lose weight in addition to reducing the risk of developing type-2 diabetes. Without choosing a primary need, it's not clear what the core job is that the product should do for its users. This, in turn, makes it harder to make the right strategic decision and offer the right product.

Determining the primary need can require trading off user and customer needs. In some cases, these might even be conflicting. Take a medical device like an X-ray machine. Its users, the radiologists, will want to use the machine to create diagnoses accurately and quickly. But the customer, a hospital trust, is likely to be more concerned about low total cost of ownership. If we prioritise the customer needs over the user ones, we are likely to develop an X-ray machine that has a reasonable purchase price and that is cost-effective to service. But if the radiologists struggle to make correct diagnoses or if they spend too much time operating the machine, then the hospital trust is unlikely to order more units and will hardly recommend the product to other potential customers. As this example shows, offering a product that truly benefits its users is often the basis for experiencing sustained commercial success. I therefore generally recommend putting users first.

Describe Users and Customers with Personas

A helpful technique to describe the beneficiaries of your product is the use of *personas*.[40] Personas are fictional characters that describe target users and customers. They can make it easier to relate to the individuals. They can guide the work of the development teams and stakeholders and help design the right user experience and create the right marketing collateral, for example.

A Persona Template

To help you take advantage of personas, I have developed a simple yet effective template that consists of three sections: picture and name, details, and persona goal, as Figure 15 illustrates. Unlike traditional persona descriptions, which are usually detail-rich user models, my template encourages you to create simple and focused personas that capture the essence of the characters. You can download the template from my website, romanpichler.com.

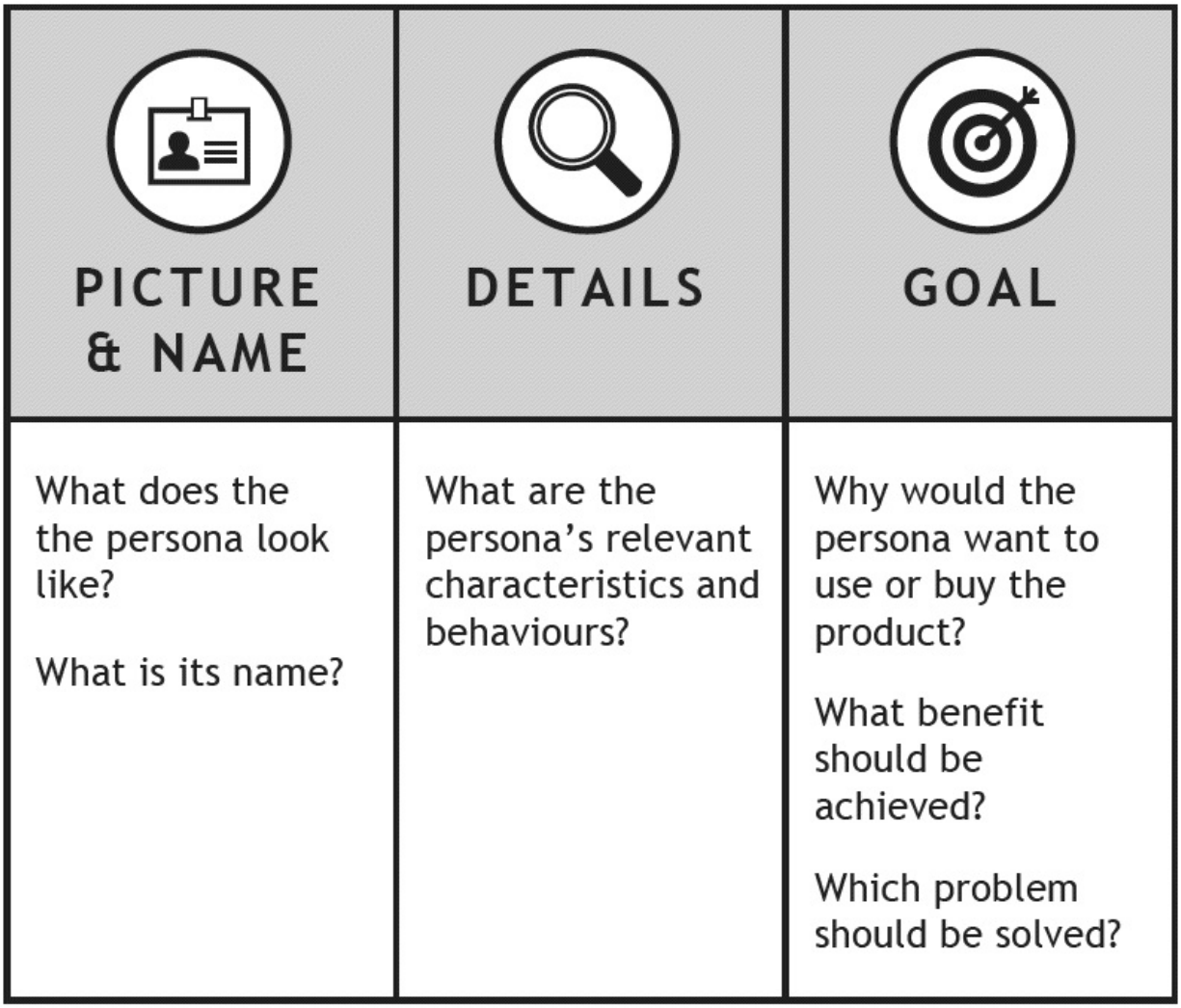


Figure 15: Persona Template

Let's look at the template's three sections in more detail. The first one captures the *picture* and *name* of the persona. Giving your character a name and a face makes it easier to relate to the character and to develop empathy for the users. It also allows you to include the persona in user stories, as I recommended earlier. Make sure, though, that you choose a representative name and picture that makes the persona believable. Say that I am creating a character that should represent teenagers with eating disorders, and I call it Gertrude. While that's a perfectly fine name, it is not very common amongst young people in the UK at the time of writing. Consequently, I should consider choosing a different, more realistic first name for my persona.

The second section lists the relevant *details*, the characteristics, attitudes, and behaviours of the persona. This can include demographics, job-related information, and hobbies. Don't make the mistake of listing everything that might be relevant. Focus on the

details that are important to understand the character. If, for example, a demographic attribute such as age or job role is not necessary, then leave it out. This helps you keep the persona descriptions concise and easy to understand.

The third section states the persona's *goal*. That's the problem that the persona wants to overcome, the benefit the character wants to gain, or the job it wants to get done. Describe the goal from the persona's perspective. Don't make the mistake of setting it based on what you think your product should do, or what it can do today. Base it instead on the research you have carried out. Additionally, make the goal specific and state it clearly. While it's fine to list more than one problem or benefit, I recommend that you focus on the main reason for the persona to buy or use your product, as I discuss in more detail below.

Tips for Creating Effective Personas

The following seven guidelines will help you develop realistic and helpful personas.

First, *get to know your audience* and understand their needs before you create any personas, for instance, by observing and interviewing people. This ensures that your characters accurately represent the target group. It avoids the risk that they are based on ideas and speculation rather than first-hand knowledge. If you have created empathy maps as discussed in the previous section, then use them to identify the right characters, for instance, by combining several maps into one persona description.

Second, *cocreate the personas* with the development team members. This will make it more likely that the dev teams understand the characters and use them to design and build the product.

Third, *distinguish between user personas and buyer personas*, as their goals and characteristics may significantly differ. This is particularly helpful for B2B products like enterprise software or health-care equipment. Take the example of an X-ray machine introduced in the previous section. While the radiologists will want to create accurate diagnoses, a hospital trust that purchases the product will have a different goal, for instance, low total cost of ownership.

Fourth, *select a primary persona* once you have created a cast of characters. This is the persona you mainly develop the product for. Working with a primary persona creates focus and facilitates decision making: the goal of the primary persona should largely determine the user experience and the product's functionality. If you find it difficult to choose one primary persona, then this may indicate that your target market is too large and heterogeneous, or that your product has become too big and complex. If that's the case, then consider re-segmenting the market and unbundling your product or introducing product variants, as I'll describe later in this chapter.

Fifth, *focus each persona on one main goal* instead of using a lengthy list of objectives. Capture the primary reason why the persona would want to use the product. This creates clarity and focus, and it facilitates choosing the right user experience design and product functionality. If you believe that the other goals are too important to be omitted, then prioritise the goals and move the primary one to the top.

Sixth, *start with initial, good-enough persona descriptions*. Then iterate over them and adjust and refine them. Don't make the mistake of trying to create perfect personas. Remember that the characters are means to an end: to design and build the right product and to create the necessary complementing artefacts like the marketing and sales materials.

Finally, *put your personas to work*. For example, leverage their goals to discover the right product functionality and include the persona name in your epics and user stories. Say that I've created a persona called Bob for my healthy-eating app. I might then capture the following story: "As Bob, I want to easily access my calories intake of the last seven days so I can understand if I have eaten fewer calories."[41]

Make Your Product Stand Out

Few products are ground-breaking innovations with no competition. Chances are that alternatives for your product do exist. You should therefore ensure that your product stands out from the crowd and that people choose your product over competing offerings. This requires you to understand who your competitors are and how your product scores against their offerings. In this section, I discuss three tools that will help you differentiate your product from its competitors: the strategy canvas, the Kano model, and the Eliminate-Reduce-Raise-Create (ERRC) grid.

The Strategy Canvas

A great tool to help set your product apart from the competition is the *strategy canvas*. The canvas forms part of the Blue Ocean Theory (Kim and Mauborgne 2004). It was originally developed to rank a business against its competitors and discover new growth opportunities, which the theory refers to as blue markets or oceans. Fortunately, the tool can also be used for individual products, as Figure 16 shows.

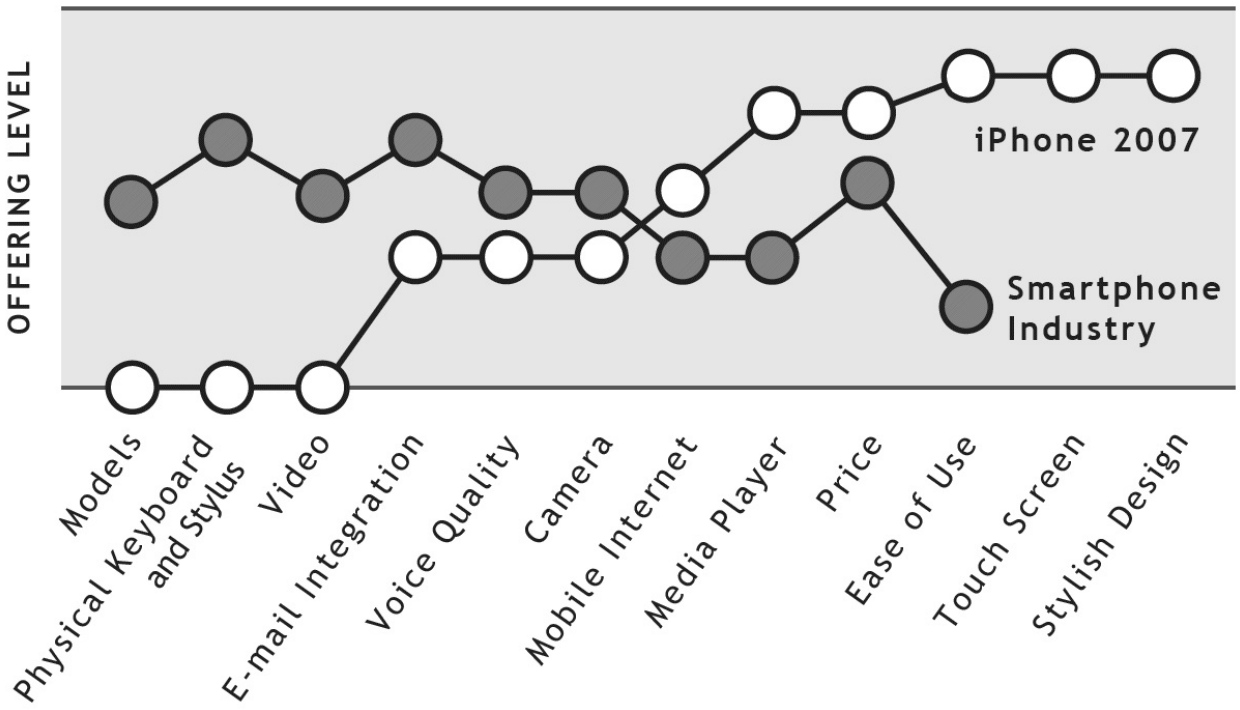


Figure 16: Strategy Canvas for the First iPhone, adapted from Islam and Ozcan (2012)

The strategy canvas in Figure 16 ranks the first iPhone against its rivals like the Nokia N95 and the BlackBerry Curve. The horizontal axis of the canvas captures the key factors companies compete on to provide their products. In Figure 16, there are twelve factors, which range from offering different models to having a stylish design. The vertical axis of the strategy canvas describes the offering level, the degree to which the competitors offer the factors. Video, for instance, was not offered on the original iPhone but it was provided by its competitors; a camera was available on the first iPhone, its quality, however, was inferior compared to the competition; but its media player was superior. Additionally, the iPhone offered two new factors, a touch screen and a stylish design. By assessing the degree to which the iPhone and the competitor products offer the twelve factors, two lines are created. The dark one represents the value curve of the smartphone industry in 2007; the light one describes the first iPhone. When we compare the two lines, we see that they diverge. This tells us that at its launch, the iPhone was clearly differentiated from its competitors. Apple achieved this by removing, reducing, and improving features as well as creating new ones. For example, physical keyboard and video were removed and not offered; voice quality and email integration were reduced; mobile Internet and MP3 player were improved; and a touch screen and a new, stylish design were created.

To apply the canvas to your product, first determine the key factors. Make sure that you choose those factors that define the current standard in your market and are used to advertise and sell products, for example, rather than the ones that favour your own product. Product reviews and test reports can help you with this, since they assess a product against the expected standard. Additionally, limit the number of factors you use

to about ten. This creates focus and avoids an overly complex canvas. With the key factors in place, rank the competing offerings and your own product by considering the degree to which they fulfil each factor. This is not meant to be a scientific exercise but based on the information you have collected whilst researching the factors. Consider if a factor is offered not at all, hardly, to some extent, or fully, and rank the products accordingly. With the scores in place, represented as dots or circles, connect them to create the value curves. Consider combining the competitor curves into a single one, like I did in Figure 16. This simplifies the canvas, and it makes it easier to discover where the main opportunities are for making your product stand out. If the value curve of your product is too close to the curve of the competition, then you haven't differentiated your product sufficiently. You will subsequently find it hard to explain to your users and customers why they should choose your product. What you would like to see instead is a value curve that clearly diverges from the industry standard, like the white-dotted one in Figure 16. This is achieved by eliminating, reducing, and raising the appropriate key factors, and by creating new ones. The *Eliminate-Reduce-Raise-Create grid* discussed below can help you with this.

The strategy canvas is not only helpful for brand-new products. Any product that hasn't entered the decline stage has to be adequately differentiated. You should therefore check on a regular basis if your product still stands out from the crowd and if the competition has caught up with it. Take the iPhone as an example. If we consider how the smartphone market has changed since the launch of the iPhone, we see that the competitors have matched the original iPhone's features. Trying to stay ahead of the competition, Apple has introduced and raised several factors over the years, for instance, face recognition, additional camera lenses, and enhanced video capabilities.

The Kano Model

Another tool that helps you differentiate your product is the *Kano model*, first described in Kano (1984) and depicted in Figure 17.

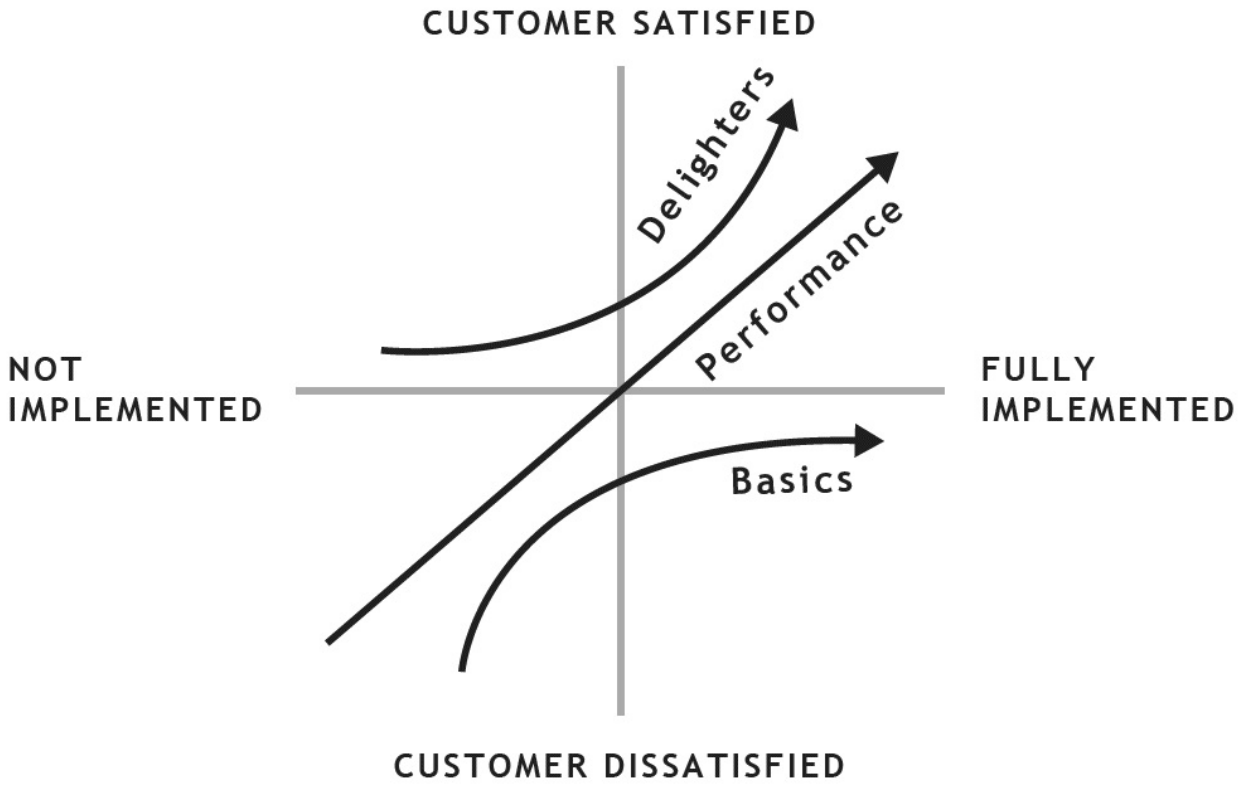


Figure 17: The Kano Model

The Kano model looks at two dimensions: the degree to which a feature is provided, shown on the horizontal axis, and the resulting customer satisfaction, depicted on the vertical axis. This allows us to distinguish three different feature categories: *basics*, *performers*, and *delighters*.[42] Basic features are must-haves; without them you typically cannot sell your product. Making phone calls and sending text messages were basic features of the first iPhone, for instance. Performance features lead to a linear increase in satisfaction. They follow the principle "The more, the better." Think of the Internet browser and a media player the original iPhone offered. Performance features, however, are not sufficient to differentiate your product in the marketplace. What you need are delighters. As the name suggests, these features delight or excite customers. The touch screen and the new stylish design were delighters of the first iPhone, for example.

The Kano model makes an important prediction: Delighters will become performers over time, and performers will turn into basics. In other words, the features that help differentiate your product today won't ensure its attractiveness in the future. Take, for instance, the touch screen, one of the delighters of the original iPhone. Today, it's a basic feature every smartphone offers; it no longer makes a phone stand out. You therefore have to invest in keeping your product attractive and properly differentiated. You can achieve this by eliminating, reducing, raising, and creating features.

Kano Analysis

To apply the Kano model, you would traditionally devise a questionnaire and conduct interviews with prospective customers and users. You might ask people, for example, how they would feel if the product had a certain feature and what they would think if the product lacked it. You might also ask how they felt if the feature was offered to a greater and to a lesser extent.

This approach differs from what I have recommended above, which is an informal application of the model in order to understand if your product possesses delighters or standout features that will sufficiently differentiate it from competing offerings. This is not to say, however, that you could or should not use the Kano model to validate the attractiveness of product features with target users and customers. If you choose to do this, I recommend using prototypes to collect user and customer feedback—instead of asking people about features which they may or may not be able to relate to without seeing them. This is likely to provide you with better, more reliable data.

Additionally, don't forget that you will validate most of the product functionality once you have moved into product development and started to build the actual product—assuming that you use an empirical, iterative process like Scrum. Therefore, apply the model only to features that are likely to have an impact on the product's desirability and ignore all others for now.

The Eliminate-Reduce-Raise-Create Grid

When you compare a new product to the competition and determine what makes it special, it can be tempting to say: "Our product must provide all the competitors' features, be better, and offer more!" Similarly, as your product grows and matures, you may be tempted to add an increasing number of features to keep it competitive. Unfortunately, this can lead to an overly complex product that takes a lot of time and money to develop, has a vague value proposition, provides a poor user experience, and is expensive to maintain. The trick is not to blindly add features, but rather to explore which ones you can remove, thereby simplifying and decluttering your product.

A great tool for achieving this is the *Eliminate-Reduce-Raise-Create (ERRC) grid*, which also comes from the Blue Ocean Theory (Kim and Mauborgne 2004). As its name suggests, the grid consists of four elements. *Eliminate* encourages you to state features that you will not provide or remove from your product. *Reduce* contains features that will be offered to a lesser extent compared with alternatives. *Raise* lists the features that you will strengthen and enhance; and *create*, finally, identifies new features that competitor offerings lack and that will be added to the product. Let's build on the example from the previous sections and apply the grid to the first iPhone.

As Figure 18 shows, the first iPhone eliminated several smartphone features that were considered standards or must-haves when the product was launched in 2007. These included different models to choose from, a physical keyboard, and the ability to shoot videos.[43] It also reduced some features, such as voice and camera quality and email integration. However, the iPhone also provided enhanced and genuinely new features. The former included mobile Internet in the form of a native web browser and a superior MP3 player. The latter consisted of a brand-new, eye-catching design and a revolutionary touch screen.

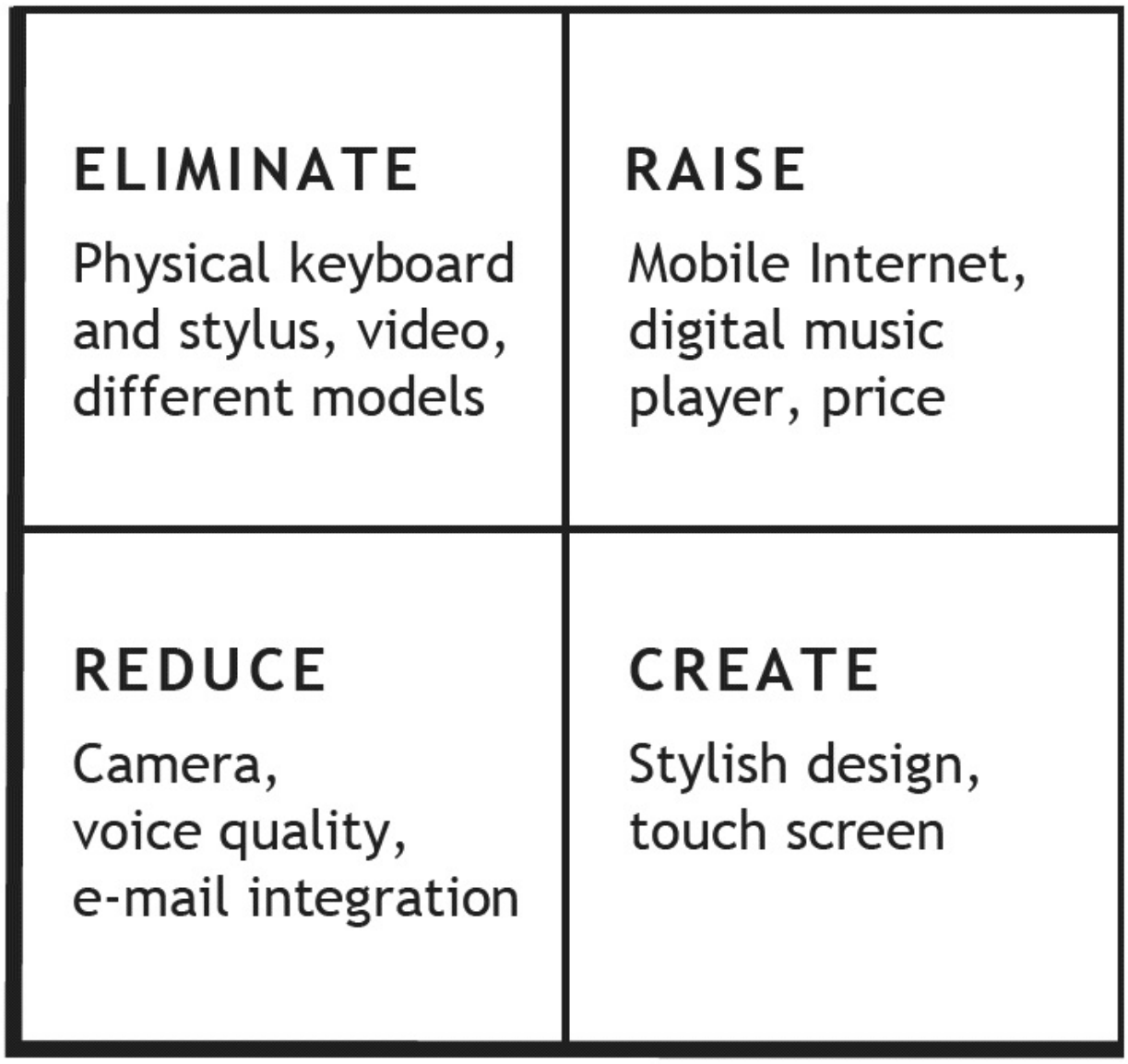


Figure 18: The Eliminate-Reduce-Raise-Create Grid

Note that eliminating and reducing the features requires a solid understanding of your target group and the problem your product solves—as well as a good portion of courage. It's easier to create a me-too product than one that stands out. Removing and weakening standard smartphone features paid off for Apple, though. It helped the company reduce time to market; it resulted in an uncluttered, easy-to-use product; and it made the new and improved features stand out.

Eliminating Features When Rewriting a Product

When you rewrite a product, it can be tempting to assume that all features of the old product must be present in the new one. But this would be a mistake. Chances are that some features are hardly used at present and

have become largely redundant. These features are prime candidates for elimination.

To identify them, carry out some user research, for instance, observe how users interact with the product. Additionally, evaluate analytics data to better understand common user journeys. If some features are only used by a specific group of users, then you might consider unbundling them and releasing them as a stand-alone product, as I discuss in more detail in the next section. This will result in more focused products that are better at serving their respective users and that are easier-to use than one large, monolithic offering.

What's more, replace the existing product incrementally. This allows you to release early product increments and initial versions of the new product to a test group, collect early user feedback and data, and find out if a simplified product with fewer features does a good job for the users. This will also provide you with more data to show to those who are fond of the features shortlisted for elimination that the functionality is not required.

Capture Your Strategy with the Product Vision Board

The best product strategy is useless if you can't clearly express and communicate it. To help you with this challenge, I have developed the *product vision board*. The board, shown in Figure 19, consists of five sections: vision, target group, needs, product, and business goals. The top one captures the vision; the bottom four describe the product strategy.

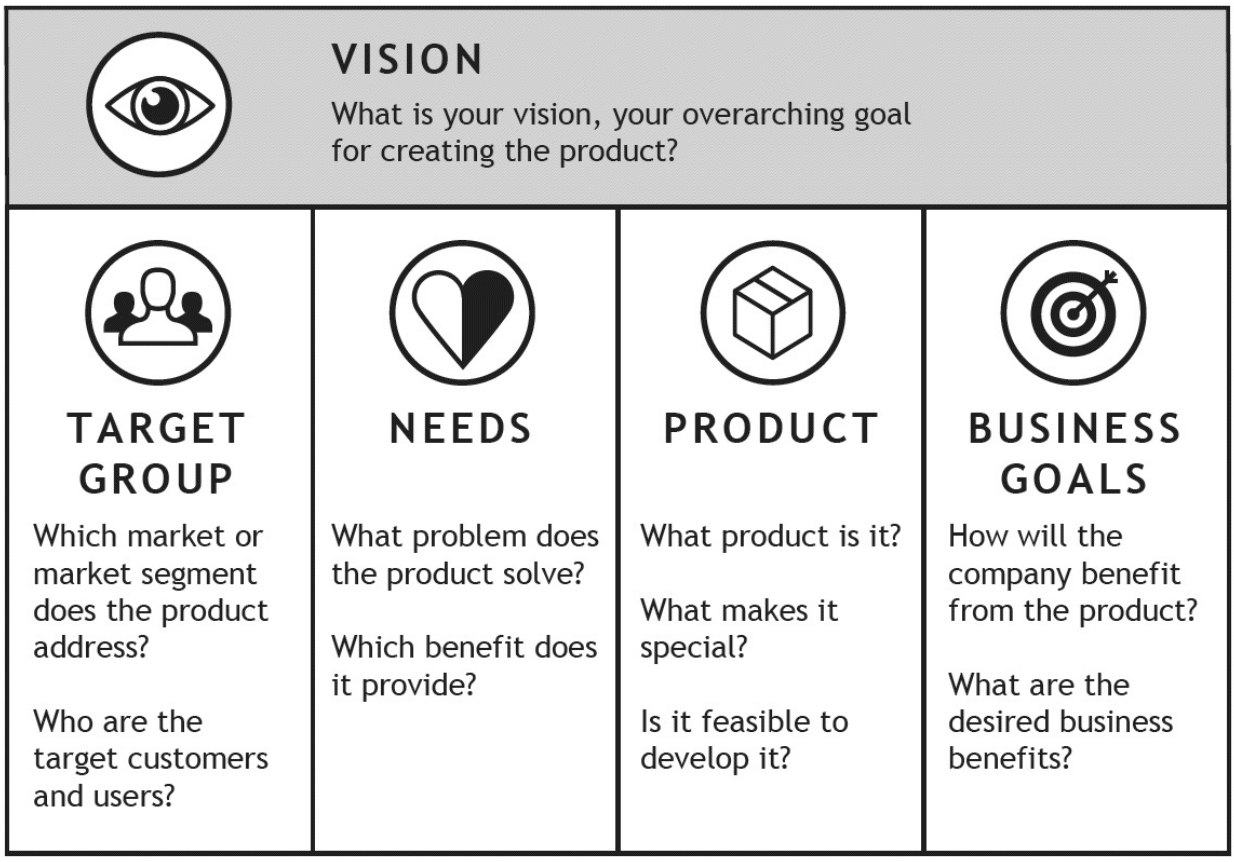


Figure 19: The Product Vision Board

In the following paragraphs, I'll share tips for working with the product vision board, which you can download from my website, www.romanpichler.com.[44]

Let the Vision Guide You

As mentioned before, the product vision captures the ultimate purpose for creating the product. Since it is truly fundamental, I recommend starting with the vision section describing the product vision first. Take advantage of the advice I shared earlier: Choose a vision that is inspiring, shared, ethical, concise, ambitious, and enduring. Make sure that it resonates with the stakeholders and development team members, avoid the mistake of restating the product idea or a business objective, and select a big vision that guides everyone for the next five to ten years.

Focus on a Specific Market or Market Segment

Describe the target group so that you can clearly tell if someone is included or not. This usually entails using demographic and behavioural attributes, as I discussed in the section *Segment the Market*. Avoid working with a big and diverse target group, for instance, "people who own a smartphone." Instead, choose a specific segment using the segmentation techniques discussed earlier in this chapter. Otherwise, you will find it hard to validate that you have selected the right group. What's more, coming up with a compelling value proposition and creating an attractive offering will be hard. The larger the target group is, the more feature-rich the product tends to be. Finally, if the users and customers are distinct, then capture both groups separately. Order them to indicate which one takes priority. For example, putting the users above the customers communicates that the users come first and that their needs are more important than the customers'.

Clearly State the Main Problem or Benefit

When you work on the needs section, ask yourself why anyone would want to use or pay for your product. Which problem will your product solve? Which pain or discomfort will it remove, or which benefit will it create? Will the product truly benefit its users or at least not harm them in any way? Be specific and avoid generic statements such as "lose weight," to stay with the healthy-eating example. Instead, describe the actual benefit someone wants to achieve by losing weight and try to quantify or qualify it, for instance, "reduce the risk of developing type-2 diabetes by 20%," as recommended earlier.

If you identify several needs, then determine the *primary* problem or benefit, and move it to the top of the section. This focuses and directs the work of the stakeholders and development teams. I find that if I am not able to select and clearly describe the main problem or benefit, I don't properly understand yet why people would want to use and to buy the product. Consequently, I need to carry out more discovery work, for example, observe and interview prospective users.

Finally, make sure that you don't mistake product features—in the sense of product capabilities—for needs. A need describes *why* someone would want to use or pay for the product. A feature describes *what* the product should do. Statements like "is easy to use,"

"works on iOS and Android," and "counts the calories consumed" are therefore features, not needs.

Describe What Makes Your Product Special

Once you have captured the needs, capture your actual product idea. State the three to five key features that make the product desirable, that set it apart from its competitors, and that encourage the users and customers to choose it over competing offerings—without getting the users hooked or negatively impacting them in another way. Examples for my healthy-eating product might be "measure and record sugar levels in food, analyse eating habits and suggest changes," and "seamlessly integrate with leading smart scales.

When working on this section, avoid the following two mistakes. First, don't turn it into a mini product backlog. Don't describe the product comprehensively or in a great amount of detail. Rather identify those features that make it special or unique.[45] Second, don't cling to a preconceived product idea. Cultivate an open mind and be prepared to change it based on what you will learn. The objective is not to make a specific product idea work but to generate value for the users and the business.

Capture the Desired Benefits for the Business

Use the business goals section to state the product's desired business benefits. Make explicit why it is worthwhile for your company to invest in the product. Avoid the mistake of using generic goals like "make money, help the company grow, be the number one weight loss app." Such general statements make it difficult to understand if the goals are feasible and how much value the product is likely to create for your business. What's more, they'll make it hard to measure progress and determine if you have met the goals.

Iterate over the business goals until they become clear and specific. For instance, make the goal "be the number one weight loss app" more precise by describing the desired business impact, the specific benefit the company will experience, such as an increase in brand equity. This will help you select the right key performance indicators (KPIs) as well as acquire a budget if the latter is required.

Finally, prioritise the business goals and state them in the order of their importance. This will create focus, guide your efforts, and help you select the right business model.

Make your Product Vision Board Testable

Whenever you create a new product vision board or update an existing one, your goal should be to create a *testable* strategy, which you can validate. To achieve this, iterate over the four bottom sections until their contents are so specific that you can determine if a statement is correct or false. If you struggle to do this, then this may indicate that you have to carry out additional discovery work like observing and interviewing users or performing competitive analysis.

Use a Collaborative Workshop to Create the Board

A great way to create the product vision board is to invite the key stakeholders and development team members to a joint, collaborative workshop. Such a workshop helps you generate buy-in, create shared ownership, and leverage the collective knowledge and creativity of the group. What's more, selling an existing vision and product strategy can be challenging; co-creation is often the better option.

You usually don't need more than half a day to collaboratively create an initial, testable product vision board, assuming that the workshop is well prepared and effectively facilitated, along the lines described in *Collaborate with the Stakeholders and Development* Teams. If you find that half a day is not enough time, then this might indicate that you lack the relevant knowledge. In this case, pause the effort to capture the product strategy and carry out the necessary discovery work. Then get back together and continue working on the board.

Complement Your Strategy with a Business Model

It's great to determine a product strategy. But you should also understand if and how you can reach the desired business benefits it contains, and how you can monetise your product—be it by selling it or using it to sell another product or service. In other words, you should complement your product strategy with a business model.

Sample Business Models

Common business models for digital products include subscription, freemium, advertising, and bait and hook. A *subscription* business model requires customers to pay a subscription fee to access the product, as is the case for Microsoft Office and Adobe Creative Cloud, for example. *Freemium* means offering a basic version for free but charging for premium features, as Spotify and Strava do. The former is a popular music streaming product; the latter connects cyclists and runners and allows them to track their activities. *Advertising* generates revenue from in-app or online ads—a business model employed, for instance, by YouTube, Facebook, and Instagram at the time of writing. *Bait and hook* provides a free or discounted product and generates revenue from selling another product or service that locks in the customer. Take, for example, iTunes around 2005: while the product itself was free, it was only truly useful when combined with a comparatively expensive iPod.

Capturing the Business Model

For some products, the business model already exists. This was the case for iTunes, to continue the example from the previous section. The product was originally offered to help sell iPods and execute an existing business model. But for other products, such as Facebook and Twitter, the product and the business model are created together. A popular and comprehensive tool to describe a business model is the *business model canvas* (Osterwalder and Pigneur 2010). But if the product vision board described earlier resonates with you, then you can use an extended version of the board. This allows you to capture your business model alongside the vision and product strategy, which can be

especially helpful when the product and business model are developed together. Figure 20 shows the *extended product vision board*.

Figure 20: The Extended Product Vision Board

The first and second rows of the extended product vision board in Figure 20 are identical to the standard version described earlier. The business model is captured in the bottom row, which provides the following four sections:[46]

- *Competitors* describes the strengths and weaknesses of the competition and their products. It builds on your insights from performing a competitive analysis and helps ensure that your product stands out.
- *Revenue sources* captures the way your product generates money, for instance, by selling licences, subscriptions, or ads, or by charging for premium features.
- *Cost factors* states the cost incurred by developing, marketing, selling, and supporting your product. This includes the cost of acquiring users and customers, purchasing third-party components, and paying for the services and products provided by partners and suppliers.
- *Channels* are the ways you will use to reach the users and customers and to sell and deliver the product to them. The latter can range from implementing an app to working with retailers to get shelf space for a shrink-wrapped product. Consider if the appropriate sales and marketing channels already exist, or if you have to create or acquire them.

You can download the extended product vision board from my website, where more information on the tool is available.

Using the Business Model to Create a Business Case

While a business model describes how a product can generate value for a company, it usually does not quantify the benefits. That's done by a business case. This plan forecasts the financial performance of a product over the coming years. A business case allows you to assess if developing the product is economically viable. Fortunately, you can derive a business case from the business model by quantifying the benefits generated and the cost incurred over the next, say, two to three years. How difficult this is, depends on the product's innovation type, as discussed earlier. For a core product, creating a realistic business case is usually feasible; for an adjacent product, it can be hard. But for a disruptive product, it is usually impossible, as the market for the product does not exist yet. You can nevertheless use the business model to justify the investment together with the inaction risk, the risk of not taking action and not being able to attain the anticipated business benefits.

Consider Your Product's Ethicality

Digital products can profoundly affect users, from enriching people's lives to exposing them to harmful content. It is therefore important that you carefully consider the impact of your product and make ethically sound product decisions.

An *ethical product* is an offering that does not cause any harm, neither to the users nor the planet. Such a product does not negatively impact people's mental wellbeing, and it

does not contribute to pollution and climate change by how it is developed, provided, and —if it includes hardware and plastics—manufactured, delivered, and disposed of. The four guidelines below will help you create an ethical product.

Users First

Putting users first may sound like a no-brainer. Sometimes, however, we take a rather superficial look at the user needs and forget to properly investigate how using the offering can affect people's mental wellbeing. Is it right, for example, to offer a product that the users hooked and creates addictive behaviour even if people want to use it? Or is it acceptable to enable the distribution of content that receives plenty of views and comments but contains material that promotes misinformation, self-harm, or violence? Personally, I don't think so.

It is our responsibility as product people to care for the users' mental wellbeing and mitigate the impact our product has on it. This does not require a degree in psychology. Taking a real interest in the users and cultivating a warm-hearted attitude towards them is usually enough. To do so, follow the advice shared earlier in this chapter. Make an effort to talk to the users, attentively listen to their ideas, and observe how they interact with your product. This helps you empathise with the individuals, understand the impact the product is having or is likely to have on them, and discover opportunities for enhancing the offering.

Fair Business Model

When digital products are offered for free, monetisation often happens by exposing users to ads and selling their data. But this approach only works if enough people sufficiently engage with the product. It can consequently be tempting to encourage unhealthy behaviour like getting users hooked to an app, which then impacts the users' mental health. The solution, in my mind, is to change the underlying business model and ensure that it is fair to all parties. This may mean moving away from monetising digital products through ads and data sales. Consider, for example, what has happened in online media in recent years: More and more online content is offered as pay-for. This includes online news articles as well as music streaming services. What's more, if a product has a compelling value proposition, then you should be able to monetise it without offering it (entirely) for free.

Right Design and Technology Choices

Ask your development team to make ethically sound decisions when designing and building the product. Discourage unethical practices like the use of dark patterns (Brignull 2021) and suggest the application of calm technology (Case 2015). Dark patterns make users do things that they didn't want to, like buying a product or signing up for a service. Using calm technology means offering products that are non-intrusive and stay at the user's periphery instead of trying to be at the centre of attention. This may mean, for instance, using notifications sparsely.[47]

Additionally, encourage the team to mitigate algorithmic biases and design for fairness when using machine learning technology. If the data that is used to train the algorithms is biassed, then your product's recommendations will be biassed too.

As design and implementation work can be absorbing, it is easy to forget about ethical design and technology choices. To mitigate this risk, include appropriate criteria in the *Definition of Done*. This definition describes the standards a product increment must fulfil at the end of a sprint. It's an important artefact that is used in Scrum.[48]

Environmental Impact

Last but not least, consider the impact your product has on the environment. Even though it might be purely digital, your product still consumes energy when it is developed and hosted, which might contribute to global warming. Therefore, choose a carbon-neutral provider who can host your digital product in an environmentally sustainable way. Additionally, reduce the amount of travel that takes place to develop the product. Personally, I've found that video calls can replace most business trips. That's not only good for the planet; it also helps your company save money, and it allows you to sleep in your own bed.

Develop Variants and Unbundle Your Product

In many ways, a product strategy wants to help you move a product as quickly as possible into the growth stage and keep it there for as long as possible. But growth does not only mean that a product generates more and more business benefits. To sustain it, the product usually has to serve an increasingly large and heterogeneous audience. At the same time, it has to offer new and enhanced features. This can dilute its value proposition, impact the user experience, and increase the time and effort required to extend it. Creating product variants and unbundling your product are two powerful techniques that help you increase the benefits your product provides.

A *product variant* is a variation or specialisation of a product. Take, for instance, Microsoft's diagramming tool, Visio, which the company offers in two variants at the time of writing: Visio Standard and Visio Professional.[49] Or consider YouTube, which is offered in different variants. YouTube Premium, for example, offers ad-free content, video downloads, and background playback as well as access to YouTube Music, Google's music streaming service; YouTube Kids is a version aimed at children with curated content and parental control features. Creating product variants establishes a product line, or a collection of related products, which is also called a product family.

Unbundling your product means promoting a feature or feature set to a new product. A good example is Facebook Messenger, a software application that allows people to chat with their friends. Facebook took the messaging functionality originally included in its mobile app and released it as a stand-alone product in 2014. Another example is Apple's iTunes. The app initially allowed people to purchase digital music and save it on their iPods. But based on its success, it attracted more and more features over the years. It also

allowed users to watch videos, listen to podcasts, read Apple Books, and activate, manage, and backup their iPhones. With the release of MacOS Catalina in 2019, Apple unbundled iTunes. The company divided the product into separate Music, TV, and Podcasts apps.

Benefits

Creating product variants and unbundling features can offer the following five benefits.

First, it can lay the foundations for future growth, as in the case Facebook's Messenger app. Unbundling the messaging feature from the main Facebook app streamlined the product, and it allowed the company to turn the Messenger app into a platform by adding new features, such as sending money to friends and communicating directly with businesses.

Second, creating specialised products help you better serve an existing audience and address a new target group. Take iTunes as an example. Unbundling the product into three new apps as described above, has enabled Apple to offer new focused products that each have a clear value proposition and offer an improved user experience.

Third, unbundling a product and introducing one or more variants can increase the business benefits and create new revenue streams. Microsoft charges a premium for Visio Professional compared to the standard version, for instance. Similarly, Strava Premium subscriptions are a major revenue source for the company, and Facebook Messenger, with features like sending money, has generated a new revenue source for the company.

Fourth, both techniques improve your ability to respond to market changes and increase your product's competitiveness. For example, Google unbundled Google Drive into separate Docs, Sheets, and Slides apps. This allowed the company to better compete against Apple's and Microsoft's productivity tools.

Finally, smaller, more focused products require fewer people to manage and develop them. This avoids the challenge of scaling the development effort and coordinating a larger number of teams; it reduces overhead and cost; and it speeds up development.

Pitfalls to Avoid

When you unbundle a product and create new variants, watch out for the following three mistakes: First, don't offer too many specialised products, as this may leave users and customers confused or even frustrated. Review and adjust your product lines regularly and eliminate individual products when appropriate. Microsoft removed Visio Technical from its Visio product line in 2002, for example, and, as mentioned earlier, Apple discontinued the iPod Classic in 2014, thereby streamlining the iPod product family.

Second, check if a newly created product would cannibalise an existing one. Take the iPhone 6 Plus as an example. While the variant helped Apple fend off competition from other smartphone manufacturers, the product cannibalised another offering of the company—the iPad Mini.

Third, proactively manage a product family and coordinate the variants and unbundled products it contains. For instance, it would be undesirable if Apple Music, TV,

and Podcasts offered different user experiences thereby making it hard for people to switch between them. Additionally, manage dependencies between the family members and consider aligning release dates. Take Microsoft Office as an example. New versions of the Office products are usually released at the same time.

Product Strategy for Variants und Unbundled Products

When you take an existing product and turn it into several variants, I recommend that you use one overarching product strategy for the newly created products—at least initially—but separate product roadmaps for each. This keeps the variants aligned while allowing them to move ahead at different speeds following their own product goals. But products that resulted from unbundling one or more features usually benefit from having their own product strategy as well as their own product roadmap.

Take Advantage of Software Platforms

A software platform is a collection of software assets that several products use. Platforms come in two flavours: closed, internal ones that support end-user facing products and open ones that allow third parties to integrate with them. Take Amazon Web Services (AWS) as an example. The platform offers cloud-based services on which other products can be built, for instance, Netflix' video streaming. In the following paragraphs, I discuss the benefits and drawbacks of platforms, and I share tips to help you take advantage of them.

Benefits and Drawbacks of Software Platforms

Platforms can help you grow a product portfolio faster and cheaper, create a seamless user experience, and increase revenue. Let's look at the three benefits in more detail by using a product suite like Microsoft Office. If the teams developing the different productivity tools all created their own user-interface code, there would be considerable code duplication, added development cost, increased development time, and possibly a fragmented user experience when switching between the apps. A software platform that standardises the user interaction and offers additional shared services, like saving and opening files, avoids these issues. It creates a consistent user experience across the apps, and it allows the app development teams to focus on creating product-specific functionality, rather than having to develop infrastructure code. What's more, opening a platform to other companies, like Amazon did with AWS, can create a new revenue source and help diversify the business. In 2020, Amazon Web Services generated a revenue of over $45 billion.[50]

Despite their benefits, platforms come with potential drawbacks. I have seen software platforms that grew so big over time that they became a bottleneck and slowed down the rate at which the end-user facing products could be progressed. I have also witnessed a platform being retired shortly after its launch, as it did not meet the needs of the development teams who should have used it. As these examples show, it is worthwhile to

carefully consider if a platform can benefit your business and to select the right approach to build and manage it.

Platform Tips

The following three tips will help you avoid the drawbacks mentioned above and take full advantage of software platforms.

Treat the platform as a product

While a software platform is a piece of technology, it's beneficial to view it as a product in its own right, albeit a technical one. Consequently, a platform should have its own product strategy and product roadmap, key performance indicators (KPIs), product backlog, and well-designed software architecture that leverages the right technologies.

Make sure, though, that the platform decisions are guided by the needs of the products it serves. After all, a platform exists to help teams build better products faster and cheaper. In other words, the platform strategy and roadmap should be guided by the strategies and roadmaps of the products that are built on it, and its architecture should be designed to make it as easy as possible to use its services.

A software platform, therefore, is more than a set of APIs. It typically requires additional assets that allow the teams to take advantage of the platform. This includes documentation, but it may also comprise tools to code against the platform interfaces. Take AWS as an example. At the time of writing, the platform comes with a cloud development kit that supports .NET and Java and that integrates with a range of software development tools (IDEs).[51]

Start small

When you create a new software platform, start small and build a *minimum viable platform* for a small number of products—say, two to three. This allows you to release a first version of the platform comparatively quickly. Once you have shown that the platform creates the desired value, you can start expanding its services and move additional products onto the platform. This avoids the risk of launching an overly ambitious software platform that fails to provide enough value to its users, as I discuss in more detail in the next paragraph.

Take Amazon Web Services again. AWS started in the early 2000s as an internal software platform to facilitate Amazon's growth and allow its development teams to focus on customer-facing innovation. After succeeding with this objective, Amazon decided to open up the platform. The company launched its first cloud-storage service that third-party developers could use in 2006. At the time of writing, AWS offers more than 200 services, and it generates a significant part of Amazon's annual operating profits.[52]

Be user-led, not technology-driven

Technical products, like a software platform, can have a dark side: As they are developed by specialised teams, they might end up being over-engineered and over-complicated. In

the worst case, a platform beautifully works on its own, but using its interfaces it is difficult and the overall performance is poor. There are two techniques that can help you mitigate these risks:

1. Start developing a new software platform by staffing the platform development team with members from the product development teams.
2. Validate the platform early and frequently. For example, invite the product development teams to participate in the platform sprint reviews and ask them to share their feedback. Additionally, release early platform increments and ask the dev team members to code against them.

Employing these techniques helped create a new telco software platform I was involved with. They ensured that the platform did a great job for the products it served, and it increased its acceptance amongst the product development teams.

Create a Product Bundle

As its name suggests, creating a *product bundle* means combining separate products into a newly formed, bigger offering. For example, Microsoft Office is a software bundle that includes Word, Excel, and PowerPoint. You can't licence or subscribe to Word individually—only in combination with other Office products.

Benefits

Creating a product bundle offers three benefits. First, it is helpful when the individual products are too small or not attractive enough to succeed on their own. Take YouTube Premium. As mentioned before, the product bundles ad-free content, video downloads, and background playback as well as access to YouTube Music to make it more attractive to users to sign up and pay for the service.

Second, bundling can help increase sales—think of McDonald's Happy Meal, which gives you a hamburger, French fries, and a drink for less than buying the items individually. The idea is that customers spend more money and buy more products if they receive a bundle discount.

Finally, a product bundle can give you an advantage over the competition. Combining Internet Explorer and Windows helped Microsoft win the first "browser war" in the late 1990s, for instance. The company used the popularity of its operating system to significantly increase its browser market share to the detriment of its arch-rival, the Netscape Navigator.[53]

Pitfalls to Avoid

While bundling can be a helpful strategy, you should avoid the following three mistakes. First, don't turn people off by creating bundles that are too big. When Netscape came under pressure in the first browser war, the company tried to make its product more attractive and defend its market share by adding several products to its browser, including

an email and a news client. The resulting bundle was called the Netscape Communicator. Unfortunately, it was bloated and cumbersome to use, which added to Netscape's worries rather than alleviating them.

Second, don't restrict the buying choices of your customers too much. Take Microsoft Office as an example. Customers can only subscribe to the entire suite, not to individual products like Microsoft Word. This might prevent people who only need access to selected products from making a subscription.[54] A more flexible bundling approach is chosen by Adobe's Creative Cloud. At the time of writing, customers can subscribe to individual apps as well as a bundle that contains all products at a discounted price.

Finally, don't forget to harmonise the user experience across the products contained in the bundle to make it easy for users to switch between products. As you would expect, the key functions of the various Office products, such as opening and saving documents, work in the same way, have the same visual representation, and are found in the same place, for instance. Using a platform as described in the previous section can help you with this.

PRODUCT STRATEGY VALIDATION

Failure is a prerequisite to learning.
Eric Ries

Whenever you create a new product strategy—be it to develop a brand-new product or to make a bigger change to an existing one—it is likely to contain risks. For example, the need for your product might not be strong enough, the market you've chosen might be too small or too diverse, or the technologies required might not be available. To maximise the chances of offering a successful product, you should systematically identify and address the key risks before committing to the product strategy. In sum, you should validate the product strategy.

Iteratively Test and Correct Your Strategy

A great way to validate the product strategy is to follow an iterative, risk-driven approach.[55] Start the process by creating an initial product strategy. Then select the biggest risk: the uncertainty that must be addressed now so that you don't make the wrong strategic decisions and you don't take the product down the wrong path. Next, determine how you can best address the risk, for instance, by observing target users, interviewing customers, or building a prototype. Carry out the necessary work and collect the relevant feedback and data. Analyse the results and use the newly gained insights to decide what to do. Should you pivot, persevere, or stop? Should you stick with your strategy, significantly change it, or no longer pursue your vision? If you decide to pivot, change the strategy, and restart the validation process; if you persevere, update the plan, and select the next key risk. Figure 21 illustrates this process.

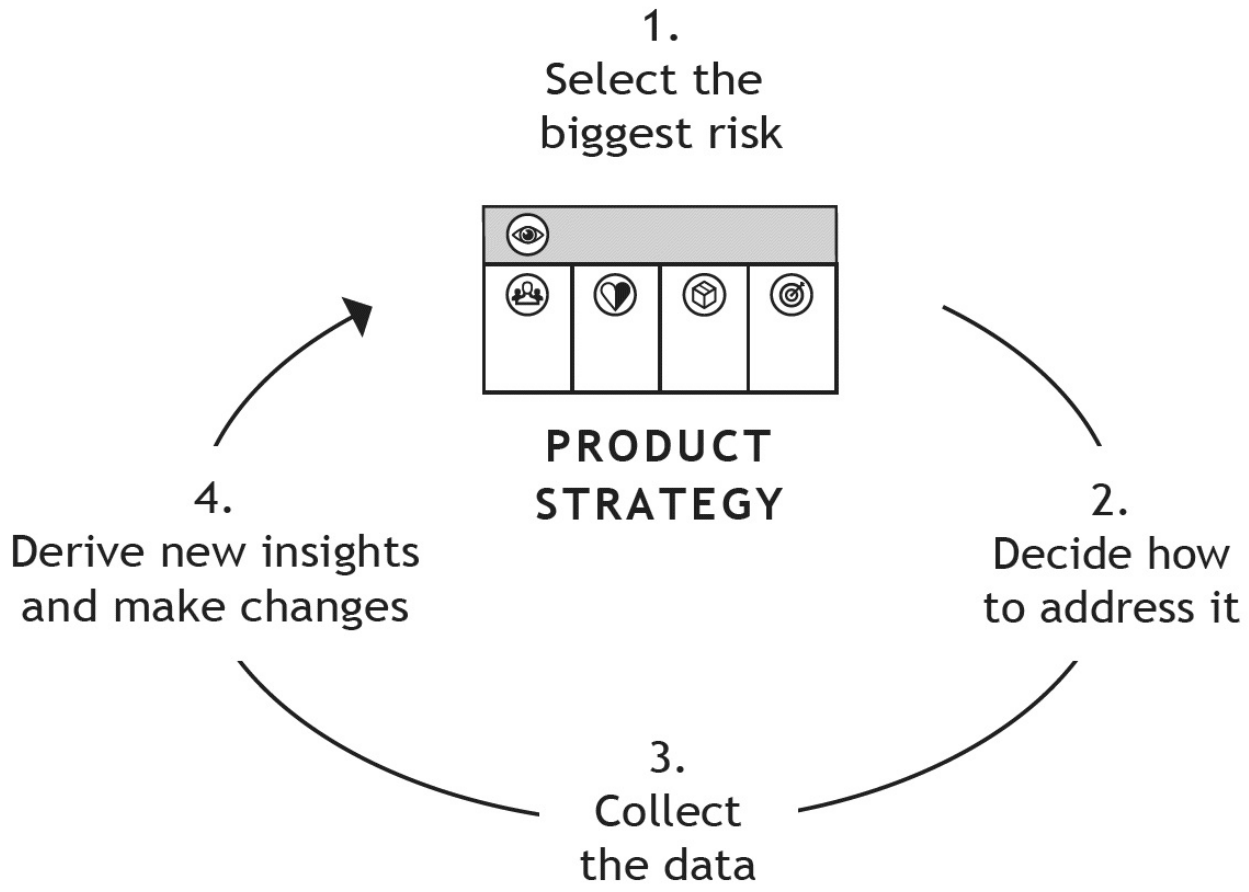


Figure 21: Risk-driven Product Strategy Validation based on Ries (2011)

Iteratively validating the product strategy offers the following two benefits: First, it encourages you to carry out *just-enough* discovery and upfront strategizing work. This avoids the risk of doing more research than is necessary—which is rather common in my experience when companies use a traditional, sequential innovation process. Second, addressing the risks early on helps you quickly understand which parts of your strategy are working and which are not. This accelerates learning, and it avoids late failure. While failing early can be challenging, it provides you with more options to change course and explore alternatives. Additionally, the cost of making the necessary changes before you have committed to implementing the strategy and started building the actual product is significantly lower than, say, finding out at launch that the need for the product is not strong enough or that the business goals are unattainable.

As you iteratively validate your strategy, you should see its uncertainty decrease; fewer and fewer risks should be present, and the content should become clearer and more refined. You have successfully validated the product strategy when it no longer contains any significant risks. You are confident that you have nailed the market, the value proposition, the standout features, the business goals, and the business model. More importantly, you have empirical evidence to support your views.

Carry Out the Minimal Upfront Discovery Work

To succeed with iterative strategy validation, you need an initial product strategy to start with. This strategy must be *testable*—the statements it contains must be specific enough so that you can identify its key risks and determine how to best address them.

When you develop a new product for a market you are familiar with and that your company already serves, you might be able to create an initial, testable product strategy straight away without having to carry out any upfront discovery work. But if you address a new market, you should carry out some initial discovery work. Take the following story from Jeffrey Liker's book *The Toyota Way*. Before the company started the development of the very first Lexus car, the person in charge of the new product, the chief engineer, was asked to spend a year in California, live the life, and drive other luxury cars. While this might sound too good to be true, there was a good reason for this approach: Toyota had never developed a luxury car before, and the company was unfamiliar with its new audience. Spending time in the target market and immersing themselves in the experience of their prospective customers allowed the chief engineer to thoroughly understand the target customers and consequently make the right product decisions.

I am not suggesting, of course, that you should necessarily spend a whole year getting to know your target users and customers. Instead, only carry out the *minimal* upfront discovery work required—just enough to be able to come up with an initial testable product strategy. How much work this will be, will depend on your current knowledge. The less you know about the target market, the more upfront work will be required. This can range from a few days to a few months. If you are in doubt, then put your current knowledge to the test and create a draft product strategy. If this does not result in a plan that allows you to identify the key risks and select the appropriate validation measures, then carry out focused discovery work until you are able to create an initial, testable strategy.

Involve the Right People

As the person in charge of the product, you are likely to find that identifying and addressing all the risks in the product strategy on your own is challenging. For example, you might not be able to spot technical risks and issues related to the marketing and sales channels. I therefore recommend employing a collaborative approach and involving the key stakeholders and development team members in the validation work. This allows you to take advantage of their expertise, and it increases the chances of effectively addressing the key risks. Additionally, it makes it more likely that the individuals understand and support the strategic product decisions. Ensure, though, that the group is stable and that its members can dedicate enough time to the validation work. Otherwise, you are likely to experience handoffs, loss of knowledge, and delays. Similarly, if the individuals don't have enough time to engage in the validation work, progress will be slow. When in doubt, weigh up a potential delay against an increase in cost, such as freeing the individuals from other duties. Consider bringing people together using an incubator, which is particularly helpful to develop the strategy of a brand-new product, as I discuss in more detail later in

this chapter. Finally, don't forget to involve the Scrum Master. The individual can be of great help to facilitate meetings and help organise and track the validation work.

A great way to get started is to run a product strategy workshop along the lines described in the chapter *Product Strategy Foundations*. The objective of the workshop is to establish an initial, testable, and agreed strategy. If you are working on a new product, you can also use the workshop to create a product vision. Once an initial strategy has been created, continue to engage the stakeholders and development team members in the validation work. The sales rep, for instance, might be the right person to address risks related to using the existing sales channels; a dev team member with a UX background might be able to prepare and conduct user interviews with you; and a developer and tester might evaluate different technology and architecture options. Using a Kanban board and weekly review meetings will help you organise the validation work and keep everyone aligned, as I explain in more detail later in this chapter.

As mentioned before, don't forget that collaboration requires leadership. As the person in charge of the product, you should actively guide the strategy validation work. Be inclusive and open-minded. Appreciate the ideas and concerns of the stakeholders and dev team members. But don't shy away from difficult conversations. Have the courage to make a decision if you can't reach agreement within a realistic time frame. Remember: Your job is not to please the stakeholders or to broker a compromise but to achieve product success.

Use Data to Make Decisions

The process of testing and changing an initial product strategy is built on the idea that collecting data is necessary to make the right strategic product decisions. This is not to say that intuition and experience are unimportant. The former is great for coming up with new ideas and options; the latter can help you correctly evaluate the data you collect. But if you make decisions purely based on what you believe to be true, then you risk making poor decisions; after all, your belief may well be wrong. This does not only happen at work, as the following example shows. One of my favourite pastime activities is riding my bicycle. One of the apps I use to record my rides is Strava. When I compare the data Strava collects to my own riding experience, it sometimes turns out that my perception is wrong—that I was not as fast as I thought, for instance, even though I tried hard. Other times, I discover that I was faster than I had assumed. If I blindly trusted my gut feeling, then this would be a poor basis for improving my performance and becoming a better cyclist.

Another drawback of making decisions based on intuition, opinions, and beliefs is that in the case of disagreements, the person with the bigger clout and the greater influence might win. While I am a big advocate of empowerment and collaborative decision-making approaches, not all product people have the necessary authority, and not all stakeholders are collaborative. In such a context, the right data will give you the chance to successfully argue against the opinion of a powerful stakeholder. Without it, the HiPPO—the highest-paid person's opinion—might win.

Don't make the mistake, though, to assume that any data-based decision will be correct. As human beings, we all have cognitive biases, which can cause us to collect the wrong data and to misinterpret it. Confirmation bias, for example, is the tendency to favour information that confirms our preconceived notions; and authority bias is the mistake of giving greater weight to the opinions of a person in a position of authority, regardless of the context. I therefore recommend that you hold your own views and opinions lightly and don't cling to your ideas. This will make it easier for you to change your mind and make the right decisions. Additionally, analyse the data together with the stakeholders and development team members. A collaborative approach is likely to balance out individual preferences and biases.

Turn Failure into Opportunity

Making mistakes and experiencing failure are unavoidable when we create something new. As Albert Einstein famously said, "A person who never made a mistake, never tried anything new." You should therefore expect to receive feedback and data that shows that your product strategy is at least partially incorrect when you start validating it. What's more, you should appreciate negative feedback and see it as an opportunity to learn and to make the right product decisions. But in practice, accepting failure can be difficult. There are two reasons for this: organisational context and mindset.

If you work for an established company that has optimised its processes and tools for maintaining existing products and consequently focuses on operational excellence and flawless execution, then you may find it hard to get people to understand that you need to fail to succeed when trying something new—it can feel like swimming against the current. Additionally, if you view yourself as a knowledgeable product management professional and you identify yourself with what you know and what you have achieved, it can be difficult to accept that some of your ideas are wrong. The solution therefore is to create the right organisational context and to embrace the right attitude.

A great way to do this is the use of an incubator. An incubator is a new, temporary business unit that is loosely coupled to the rest of the organisation. It offers the necessary autonomy to innovate—and to fail. As Peter Drucker writes in his book *Innovation and Entrepreneurship*, "The best, and perhaps the only, way to avoid killing off the new…is to set up the innovative project from the start as a separate business."[56]

As mentioned before, an incubator is particularly valuable for disruptive products due to the high amount of uncertainty and risk they exhibit. But incubators also benefit adjacent innovations, as the following example shows. The first Scrum project I worked on in 2004 was tasked with creating a new digital telecommunications product. The original development effort took place at Siemens, a company that is over a hundred years old and that had more than four hundred thousand employees in the mid 2000s. To create an incubator, we hired new offices and collocated people from different parts of the company. This gave the team the ability to collaborate effectively. It provided them with the freedom to think outside the box, to try out new things, and to fail, learn, and improve. Had the team members been embedded in their respective organisations, it would have been impossible to achieve the same success.

There are, of course, alternatives to using an incubator. An example is Google's 20% rule. Engineers can spend up to 20% of their time exploring new ideas, an approach that helped create the Google Chrome browser, for instance. Another alternative is a company-internal hackathon, where people come together for one or more days to try out new ideas. Facebook's *Like* button was conceived in this way, for example.[57] Whichever approach you choose, ensure that you create an environment that allows you to fail and to learn fast when working on adjacent and disruptive innovations.

In addition to establishing the right organisational context, you will benefit from embracing the right attitude and cultivating an open mind, as the following example shows. A few years ago, I had the idea to offer my product management templates as a digital product, and I was excited about the prospect of creating the product and using it to diversify my business. But carrying out the validation work—interviewing and observing product people and talking to procurement employees—showed me that successfully offering a software tool and selling it to larger companies was much more difficult than I had thought. But realising this was hard for me: I was attached to the idea, and I wanted to create the product. As this story shows, it is easy to cling to preconceived ideas even if we receive data that invalidates them. What helps me is not to take ideas personally and not to identify with them, but to hold them lightly, and to tell myself that they are not *my* ideas but simply thoughts and assumptions. The software tool story ended with me killing the innovation effort while validating the product strategy. This wasn't nice at the time, but it was the right thing to do. It freed up time and allowed me to write the first edition of this book.

Get Out of the Building

Whatever you do to test your product strategy, follow Steve Blank's advice and "get out of the building."[58] Visit your target users and customers to understand their needs and to see the environment where your product will be used—for instance, on the train, in a supermarket, at people's homes, or at their workplaces. If in-person interaction is not possible, then connect with the users and customers via online tools and conduct online interviews. Note that "get out of the building" is not only applicable for brand-new products, but also for existing ones. I generally recommend that you directly interact with users and customers at least once per quarter, even if you have plenty of analytics data available. This ensures that you empathise with the individuals and develop a deep understanding of their needs. This, in turn, puts you in a better position to make the right strategic decisions for your product.

While colleagues can be great sources of user and customer knowledge, you should not solely rely on what others tell you. What's more, don't blindly trust a market research company, as the following example shows. A client of mine, a large media business, spent a lot of time and money on developing a new app. After its launch, the company discovered that users only employed a fraction of its features. What went wrong? The company had hired an agency to carry out the research work and validate the strategy. For whatever reason—be it fear of failure or an insufficient understanding of the product's value proposition—the agency reported back that all the feedback on early

prototypes was positive, and that the strategy was just right. Had the research and validation effort not been outsourced and had the person in charge of the product directly interacted with the product's target audience, then a less feature-bloated and cheaper product would have most likely been created.

Finally, don't be put off by the effort required to contact people. Recruiting a test group can be as easy as sending out a targeted email or LinkedIn message. The reward might be a great idea for how to improve your product.

Identify the Biggest Risk

Once your product strategy is testable, you are ready to determine the key risks and then systematically address them, as I explain below.

Determine the Key Risks

To unearth the risks that are hidden in your product strategy, assess the statements it contains and ask yourself how certain you are that each statement is correct. Highlight or mark the statements that are risky and consequently have to be worked on, as I'll explain in more detail shortly.

Avoid the mistake of being primarily concerned with technical feasibility—something I have seen teams do especially in companies with a strong engineering culture. No matter how technologically advanced your product is, if it does not create enough value for its users and customers, it is unlikely to be a success. Therefore, *tackle the risks related to the target group and the needs first* as a rule of thumb. If there is no market for your product, or if the product's value proposition is weak, then coming up with cool features and amazing technologies is not going to help you.

The following paragraphs offer sample questions that help you determine the risks that your strategy may contain. Use them as a starting point; discard questions that are irrelevant and add additional ones that might be missing.

<u>Risks related to market and needs</u>

Are you confident that you have selected the right market or market segment and that you are addressing the right people? Is the target group large enough? Are you sure that the group is not too diverse? Are you able to clearly state who is in your target group and who is not? Do you know how big the market roughly is? Are there any barriers that will make it hard for you to enter the market?

Will addressing the needs make a real difference to the users and customers? Will it truly benefit the individuals, or will it at least not harm them in any way? Are people aware of the need you want to solve? If you have identified several needs, have you prioritised them and chosen a primary one? Are the needs specific enough so that you can tell if a need is successfully met and select the right key performance indicators?

<u>Risks associated with features and technologies</u>

Can you list the top three features that will encourage people to use or buy the product? Does the product offer a clear and compelling advantage over the competitors' offerings? Are you confident that the product's key features won't hook people or harm them in any other way? Is it feasible to develop the product? Are the technologies required available and are they mature enough? Does your organisation have the necessary skills to effectively use them? Are enough people with the right skills available, and if not, can you recruit them? Will you have to pay for any third-party software, and if that's the case, have you checked that the cost and the licence agreement will work for you? Are you clear on the key characteristics of the desired user experience? Have you considered ethical design and technology practices like the use of calm technology and the mitigation of machine learning biases?

Risks contained in the business goals

Are you confident that it's worthwhile for your company to develop and provide the product? Are you clear on the business goals that the product should deliver? Are the goals realistic? Are you able to prioritise the goals or at least, state the most important one? Have you chosen an appropriate business model to monetise the product? Are you confident that the business model is fair to all parties and that it will help you achieve the desired business benefits?

What about risks related to dates and development cost?

As you may have noticed, the questions above do not consider dates, or time frames, and development cost. There is a reason for this: I recommend determining dates and cost not in the context of the product strategy but based on a product roadmap that has been derived from the strategy, as I discuss in more detail later in this book. This puts you in a better position to understand to which extent the needs and business goals can be realised over the next, say, twelve months and what cost this is likely to incur.

Select the Biggest Risk

Once you have identified the risks contained in your strategy, choose the most crucial one —the risk that must be addressed now to avoid making the wrong decisions, taking your product down the wrong path, and experiencing an undesired outcome. As a rule of thumb, *work on one risk at a time*. This creates focus and facilitates collaboration. Additionally, it makes it easier to collect and analyse the relevant data and to track the progress of the validation work.

A simple yet effective way to determine the most important risk is to ask the key stakeholders and dev team reps to cast their vote. The process starts by giving each individual three votes. These might be coloured sticky dots if a physical representation of the product strategy is used; in an online context, the individuals might use their initials. Ask people to put the dots or initials next to the statements, which they are most unsure or concerned about. Then count the votes per statement. If there is a clear winner, briefly discuss the risk, along with the damage it might cause. For example, the target group may be too large and diverse, which might make it difficult to create a product with a

compelling value proposition. If there is no agreement, then carry out another round among the statements that have attracted the most votes. Stop voting when you reach consensus, and the biggest risk has been identified.

Note that this approach takes advantage of the collective wisdom of the group and uses the team members' perceptions to determine the risks and their severity—rather than trying to quantify the impact and probability of each risk. But it is fast and for the purpose of iterative strategy validation, it is usually good enough: If you misjudge or overlook a risk, you are likely to spot it in the next validation iteration.

Wash, Rinse, and Repeat

Don't forget to repeat the risk selection process once a risk has been successfully resolved. As mentioned before, the validation process continues until no major risks are left—or you've run out of time and money.

Choose the Right Validation Techniques

Once you have determined the biggest risk, take the next step, and decide how to best address it. This involves two things. First, select the right validation technique—the method that allows you to address the risk and to reduce the uncertainty. Second, choose the right test group—the people who should provide relevant feedback or data. Sample validation techniques include observing users, interviewing customers, and creating throwaway prototypes, which I describe in more detail in the next sections.

Directly Observe Users and Customers

Direct observation means carefully watching how target users and customers carry out a job. It helps you validate that you have picked the right target group and that its members are likely to benefit from your product. Using my healthy-eating app as an example, let's assume that I have chosen young, male professionals as my target group. To learn more about them and to understand if they would benefit from the product, I could study their eating behaviour and observe them buying and eating food during their lunch break, whether it is in a canteen, a restaurant, or a public space.

Observation is a powerful technique not only for evaluating ideas and addressing risks, but also for spotting opportunities and generating new ideas. Watching people brings a new dimension to understanding how individuals use your product, and why they might be struggling with some features or not employing them at all. Therefore, get out of the office and study the users in the wild. If that's not possible, then ask selected members of your target group if you can watch them remotely, for instance, by using screen sharing, to see how they carry out the relevant tasks.

As you directly observe members of the target group, focus on the risk you want to address, and be patient, nonintrusive, and open-minded. Being patient and gentle mitigates the risk that people will act differently with someone watching them—which is known as the *observer effect*. Cultivating an open mind reduces the risk that your

cognitive biases distort what you see and that you draw conclusions before you have gathered all the evidence.

Carry Out Problem Interviews

Problem interviews are structured conversations with target users and customers. They allow you to understand people's needs, how they currently get a job done, what works well for them, and what does not. This, in turn, helps you validate that you have chosen the right market and needs. Say that I am still working on my healthy-eating app, and I am not sure if healthy eating is an issue for young male professionals. I could then set up one-on-one interviews with members of the target group and ask them, for example, if they are aware of what and how much they eat; if they purposefully choose their food; if they are happy with their health and physical appearance; and if fitness and health are generally important to them.

To conduct effective problem interviews, I recommend applying the following seven tips.

- Be clear on the specific risk you want to address, carefully prepare the questions you want to ask, and don't forget to record the answers. The first two points will make it easier to guide the conversation; the latter will help you draw the right conclusions afterwards.
- Keep your interviews brief and consider limiting them to fifteen minutes. This is usually long enough to address a specific risk, and it makes it easier to find people who are willing to be interviewed. Additionally, consider offering a small incentive or thank-you token such as a voucher.
- Embrace a friendly and open-minded attitude. Listen attentively with the intention to understand. Be appreciative of the individual's perspective even if you disagree or find the person dislikeable. This avoids premature judgement, and it helps you collect the relevant information.
- Use the right conversation techniques including asking open questions, inserting intentional pauses, and summarising what you have heard, as I explain in more detail below.
- Don't mention your product. A problem interview is about validating the problem —not the solution. Getting feedback on the UX design and features, for instance, is not the objective at this stage.
- End the interview by asking the individual for feedback on the conversation and the questions you asked and for referrals, if you need to find more interviewees.[59]
- As a rule of thumb, conduct five to ten problem interviews to address a given risk.

As problem interviews are about conversing with a target user or customer, it's helpful to employ the right conversation techniques. This includes using open questions—which typically start with *why*, *how*, or *what*—to clarify a piece of information and to encourage the interviewee to share more information. In interviews related to my healthy-

eating app, I might say, for example, "Can you please tell me why healthy eating is important for you?" Or I might ask, "Please help me understand why having a certain weight is important to you."

Be careful not to ask leading questions that encourage people to provide a certain answer, for instance, "You are concerned about eating healthily and really want to do something about it, right?" This might cause the individual to give you the answer you want to hear even though the person does not agree with it. Be careful not to comment on the answers, whether in verbal or in nonverbal form—for example, by raising your eyebrows or by sighing, as this is likely to influence the interviewee.

Consider paraphrasing and summarising what you heard the individual say to check that it is right and give the person the opportunity to reflect on what was said. Use intentional pauses to reflect on what you have heard. Finally, gently redirect the conversation if the interviewee went off on a tangent. Table 3 summarises the five conversation techniques I just shared.

Table 3: Selected Conversation Techniques for Problem Interviews based on Pichler (2020)

Technique	Description
Clarify	Ask clarifying questions to ensure that you have correctly understood what was said and to encourage people to provide more information. You might say, for instance, "Can you please give me an example of what you mean?"
Paraphrase	In your own words, say what you think the speaker has said and ask the individual if you have stated it correctly.
Summarise	Sum up what the other person has said to check that you have heard correctly and to give the speaker the opportunity to reflect on what they said.
Pause	Intentionally stay silent for a moment. This allows you to reflect on what you have heard. You might want to say, for example, "This sounds important. Can we pause here for a moment so that I can let it sink in?"
Redirect	Acknowledge what the other person has said and state your desire to return to a previous topic or to move on to a new one. This allows you to get the conversation back on track, in case of going off on a tangent, and it avoids one issue becoming too dominant.

Use Product Fakes

A product fake is a mock-up that imitates the future product, or at least an aspect of it so that you can test a specific assumption *as cheaply and quickly as possible*. Here are four techniques that help you use product fakes:[60]

- Creating a *throw-away prototype* that implements one or more facets of the future, which you want to validate. The prototype might be software-based (clickable) or paper-based (analogue). For example, in the mid 2000s I helped create a new VoIP phone.[61] To test if we had identified the right key features, we built a throwaway prototype using Adobe Flash. The technology would have been completely unsuitable to develop the actual phone software. But it allowed us to implement the product fake quickly and cheaply.[62]

- Publishing a *video*, as Dropbox successfully did to test its original product idea.
- Deploying a *landing page* that people visit after clicking a link from an email or ad.
- Using a *Wizard of Oz* and *concierge* experiment where the product or service is manually provided to test whether there is sufficient need for what you want to offer.

Product fakes can help you understand if there is sufficient demand for your product, if its key features really delight the users, if your ideas about the user experience are correct, if the marketing and sales channels work, and if the product is priced correctly.

For instance, I might be tempted to choose a freemium model for my healthy-eating app. Using this approach, I might give away a basic version for free and aim to generate revenue by encouraging users to subscribe to a premium version. While the business model is popular, it carries a significant risk: People might be so happy with the basic version that they don't see the need to pay for additional features—something Strava has struggled with in the past, for example. But if the basic version is too thin, the app is unlikely to attract enough users, which will make it difficult to generate the desired revenue in the future. To address this risk, I could develop an initial pricing model and use a throwaway prototype and test if people would be willing to pay a specific amount for the product or a specific feature.

Design Sprints

A design sprint is an intense five-day period in which a cross-functional team tackles a design problem (Knapp et al. 2016). This includes creating prototypes and testing them on selected users. At the end of the five days, the team should better understand how to design the product so that it creates the desired value and addresses the user needs selected. If you adopt the strategy validation process explained in this chapter and consider using design sprints, then I recommend employing them after you have successfully shown that there is a need and market for the product and that the business goals can be achieved. This nicely sets up the design sprint work, and it avoids that you are prematurely concerned with the solution.

Build Spikes to Assess Technical Feasibility

A *spike* is another type of throwaway prototype—one that addresses a specific technology or architecture risk.[63] For instance, spikes could help me determine which machine learning framework would be best suited for my healthy-eating app and if a model-view-controller (MVC) architecture would be effective.

Addressing technical risks as part of the validation work helps you better understand if you can develop the product, and how much effort it is likely to require. That's important, as the best product idea is useless if you don't have the people with the right skills to build the actual product, if the technologies required don't exist or aren't mature enough, or if the product is simply unaffordable to build. Additionally, spikes give the dev team representatives the opportunity to start thinking about the development and test environments, and to investigate any third-party software that may be part of the product.

This helps the development team get ready for the first sprint, and it gives you a rough understanding of the licensing cost, which may affect your business goals and your business model.

While spikes can be very helpful, avoid the trap of creating a big design up-front (BDUF) for your product. Only tackle the key risks, and don't worry about the detailed design. You want to be confident that you can build the product in a reasonable time frame and on a realistic budget. But you should not make all architecture and user experience design decisions up front. The detailed designs should evolve during the development of the product.

Don't Rely on a Single Method and Separate Data Analysis from Data Collection

To finish off the discussion of validation techniques, let me share the following two tips with you. First, don't make the mistake of relying on a single method. Every validation technique has its strengths and weaknesses; none is perfect. Select the method that is best suited to address a specific risk. It would be a mistake, for example, to always rely on product fakes and prototypes and to ignore, for instance, direct observation and interviews. If you find that several techniques are equally well suited, then choose the quickest and cheapest one. For example, if you could use a paper-based and a software-based prototype, then opt for the former—assuming that it will be faster and more cost-effective to create.

Second, always separate data analysis from data collection. Avoid drawing any conclusions while you are still gathering the relevant information, as this can influence your data-collection work and cause you to make the wrong decisions. Wait with the analysis until enough data has become available.

Pivot, Persevere, or Stop

Once you have applied the appropriate validation technique, take the two steps. Review and analyse the feedback or data. Then draw the right conclusion and take the right action. Let's look at the two steps in more detail.

Review and Analyse the Data

Reviewing and analysing the data includes removing data whose quality is too poor to interpret it correctly and discarding irrelevant data. Say that you look through the notes taken during the last round of customer interviews but can't understand what an interviewee meant, then you should discard this piece of information. Similarly, if you consider that the perspective of an interviewee is an outlier and not representative for the rest of the target group, then you should ignore it. Be careful, though, not to fall prey to confirmation bias and reject data that does not confirm your preconceived ideas. Keep an open mind and see negative feedback and data as an opportunity to learn more and not as a personal failure, as I discuss in the section *Turn Failure into Opportunity*.

If you find that the feedback or data you have collected triggers difficult thoughts or emotions like frustration or aversion, then don't ignore them. It is normal to feel upset when you find out that idea does not work, that the need for a product is not strong enough, or that the business goals are not attainable—it's something that's happened to me more than once. But postpone the analysis work until the feelings have resided or at least weakened. This will help you evaluate the data as objectively as possible and draw the right conclusions from it.

Take the Right Action

Once you've looked at and evaluated the data, decide what to do next. There are three choices you have: pivot, persevere, and stop.

- *Pivot* means staying grounded in the vision but significantly changing the product strategy (Ries 2009b). Choose this option if the current strategy is unlikely to result in a successful product. Let's say that I cannot find a way to monetise my healthy-eating app, I might then decide to pivot and write a healthy-eating book instead. The new strategy would result in a fundamentally different product. But it would still allow me to realise my vision—to help people eat healthily.
- *Persevere* means continuing with the current product strategy. Make this choice if the validation work has confirmed that the overall strategy is correct, and only a smaller change is required, for example, refining the needs statement or adjusting one of the standout features.
- *Stop*, finally, means ending the strategy validation work. Select this option if you have already pivoted several times and the latest data suggests that the current product strategy is wrong again, or if there are no key risks left in your strategy—it has been sufficiently de-risked. In the first instance, stopping means recognising that you can't find a way to realise your vision and that everyone's time and energy is better used by working on a different innovation initiative. In the second instance, the validation work is complete. You are ready to move on and derive a product roadmap from the strategy.

If you have followed my previous advice and you have carried out as little upfront discovery work as possible, then you should *expect* that your strategy is wrong, or at least not entirely correct, when you start validating it. If you iteratively test the plan and you don't find any issues with it, then stop and reflect. Ask yourself if you are tackling the right risks, if you are choosing the right validation techniques and applying them effectively, if you are using the right test groups, and if you are evaluating the results objectively. This is, of course, easier said than done. In practice, we often want to receive positive feedback and have our ideas confirmed. But don't allow the desire to prove that your ideas are right drive the validation work. Instead, accept that without experiencing failure and learning from it, you are unlikely to make the right strategic decisions.

Plan and Track the Validation Work

Whenever you are likely to require more than a couple of days to address the key risks in your strategy, you'll benefit from planning, organising, and tracking the work. The following techniques will help you with this.

Timebox the Work

Correctly estimating the time required to validate a product strategy can be tricky: You initially don't know what you don't know, and new risks often emerge as part of the validation work. Luckily, there is a straightforward solution: timebox the strategy work.

A time box is a fixed period that cannot be extended. At the end of the time box, the work stops, and you reflect on what has been achieved so far. If the work has not been completed and your strategy still contains significant risks, you have two options: add another time box or stop the work.

To choose the right duration of the time box, consider the following two factors: First, determine the amount of uncertainty present. Core innovations carry few risks, and the validation effort is low, ranging from a few hours to a few days. Adjacent innovations exhibit a much higher level of risk and require a medium validation effort. You may have to spend several weeks addressing the key risks in your product strategy. Disruptive products are even riskier than adjacent ones. As a result, they require a high validation effort, and it may take you several months to develop a valid strategy.

Second, limit the time box to a maximum of four weeks, even if you're working on an adjacent or disruptive innovation and you are likely to require more time. Shorter time boxes create focus, and they make it easier to track the progress.

Use a Kanban Board

Once you have roughly estimated the validation work and determined the length of your validation iterations, take the next step. This involves identifying the tasks required to address the key risks, decide who carries out the tasks, and track the progress. A great way to do this is using a Kanban-based process and a simple Kanban board, like the one pictured in Figure 22.[64]

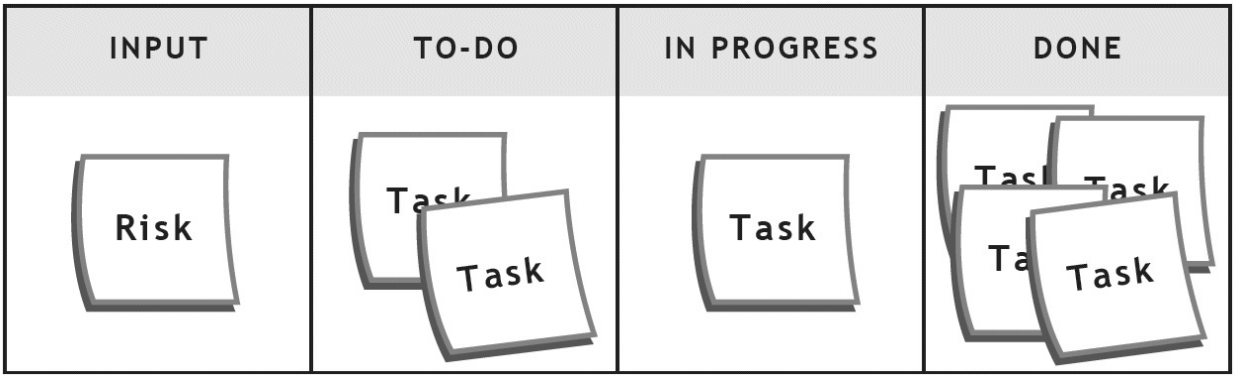


Figure 22: A Simple Kanban Board for Strategy Testing

The Kanban board in Figure 22 contains four columns: *input*, *to do*, *in progress*, and *done*. Add the risks you are working on to the input column and put the tasks necessary to address them into the to-do column. When someone works on a task, the individual moves it into the in-progress section. When the task is completed, it is moved to the done column. Say that I want to address the risk of selecting the wrong target group for my healthy-eating app. I would then capture the risk and add it to the input column of the Kanban board. Next, I would identify the relevant tasks, such as finding prospective users who are willing to be interviewed, preparing the problem interviews, conducting them, and analysing the data. Once I've added the tasks to the to-do column, I can start working on them and move them to the in-progress and eventually to the done column.

Using a Kanban board visualises the work to be done and the progress made, it facilitates collaboration, and it encourages ownership and self-organisation. That's particularly valuable when you adopt a collaborative approach and involve the key stakeholders and dev team members in the validation work, as I recommended earlier. You can improve your Kanban board by adding people's names, deadlines, and maximum effort to the tasks, if necessary. You can also introduce work-in-progress (WIP) limits to optimise the flow of work. Don't forget to ask your Scrum Master or agile coach to help you apply a Kanban-based process, especially if you are not an experienced Kanban practitioner.

Use Stand-up and Weekly Review Meetings

In addition to timeboxing the work and using a Kanban board, I recommend using daily stand-up and weekly review meetings. *Daily stand-ups* are brief meetings that allow everyone involved in the validation work to review the progress, talk about what needs to be done, and address any blockers. If you find that having daily stand-up meetings is not necessary, experiment with reducing their frequency. Hold them, for instance, two or three times per week. Then evaluate if that's sufficient or if collaboration suffers.

Use the *weekly review meetings* to assess progress and plan the work together with the key stakeholders and development team representatives. Discuss which risks have been successfully addressed and which ones are still open; determine if any new risks have emerged. Then decide what needs to be done next. This is likely to involve identifying and addressing new risks. But it might also result in a more significant change like pivoting or stopping the validation work. If you are at the end of a strategy iteration, determine if you need to carry out another cycle, and if it's duration should be the same as the current one.

You can also use the meeting to reflect on how well you are collaborating as a group and identify improvement measures, for example, securing renewed commitment from everyone to attend the stand-up meetings, involving someone from finance to address a business model-related risk, and adding WIP limits to the Kanban board to improve throughput.

Don't forget to ask the Scrum Master to facilitate the two meetings so you can focus on contributing to them rather than having to ensure that everyone actively participates and that nobody dominates.

Can I use Scrum to Manage the Strategy Validation Work?

If you use Scrum to develop products, you might be wondering if the framework can be applied to the validation work. My answer to this question is that you could use Scrum—but I wouldn't recommend it. Scrum was designed to help teams develop complex *products*. Its sprints take a *product* backlog as input, and they deliver *product* increments. But when you test a product strategy and address its key risks, you are not working on the actual product but on a plan that contains ideas. You could, of course, reinterpret the Scrum terms and replace, say, *product backlog* with *strategy backlog*. That's something I would not advise to do, however. I find it better to recognise that the Scrum is not the ideal tool for the job and use one that's better suited—which is Kanban in my mind.

PRODUCT STRATEGY REVIEWS

Just because you make a good plan, doesn't mean that's what's gonna happen.
Taylor Swift

Once you have validated the product strategy and released a product that implements it, you should measure if the desired value is being created and assess if the strategy is working. To put it differently, you should select the right key performance indicators and regularly review and adapt the product strategy, as I discuss in this chapter.

Choose the Right Key Performance Indicators (KPIs)

Key performance indicators (KPIs) are—in our context—metrics that measure how well your product is doing. They help you understand if a product is creating enough value for the users and customers, if it is meeting its business goals, and if the product strategy is working. Without KPIs, you end up guessing how well your product is performing. It's like driving a car with your vision blurred: You can't properly see if you are heading in the right direction or if you are getting closer to your destination. Using KPIs helps you balance intuition with empirical evidence. This increases the chances of making the right decisions and achieving product success. In the next following paragraphs, I'll share my advice for choosing the right indicators, I'll offer tips for avoiding common KPI mistakes, and I'll state a list of sample indicators.

Focus on Needs, Business, and Product Goals

To select the right KPIs, start with the needs and business goals stated in the product strategy and consider how you can tell that they are met. For example, if your product directly generates revenue, then monthly recurring revenue may be a key indicator. You might also want to measure customer satisfaction, loyalty, and referral rate to understand if your product is generating the desired value for its users.

Additionally, use the product goals on your product roadmap to identify further KPIs —assuming that you have derived a product roadmap from your strategy and that you have captured appropriate outcomes or goals on it. For instance, if the product goal you are aiming for is "acquire an initial user base," then you might want to choose a metric like market share to measure goal achievement.

Use Health Indicators

Measuring to which extent your product is meeting the needs, business goals, and product goals is great. But it is not enough. Say your product is achieving its revenue and profit

targets, and customer engagement and referral rates are high. This suggests that your product is performing well; there seems to be no reason to worry. If, however, the team motivation is declining or the code quality is deteriorating, you should be concerned. These indicators suggest that achieving product success will be much harder in the future, either due to an increase in absence, turnover rate, or technical debt.

You should therefore look beyond commonly used financial and customer indicators and collect the relevant product, process, and people data. This helps you understand how healthy your product and team are, spot important warning signs early on, and respond to them quickly. But this does not mean that you have to collect all the data on your own as the person in charge of the product. For example, the Scrum Master may be able to collect feedback at the end of a sprint retrospective to determine team motivation; and the dev team members may be able to supply you with code complexity data so you can track software quality.

KPIs and Technical Debt

Like a company experiencing financial debt, products can incur "technical debt." Cunningham (1992) This happens when wrong or suboptimal architecture, technology, and coding decisions are taken. Consequently, the software architecture may not be as loosely coupled as it should be, and the code may be overly complex and hard to understand.

As the person in charge of the product, you might not be terribly concerned about how clean and well-structured the software is. But the quality of a digital product directly impacts your ability to achieve product success: Technical debt makes it hard to experiment with new ideas, release new features, and quickly respond to user feedback. It is therefore in your interest to ensure that the quality of your product is adequate. This starts by selecting the right indicators and regularity measuring product quality. For a digital product, these include code complexity, architecture refactoring potential, and number and severity of open bugs.

Once you understand how soft or brittle the code is, you can take the right actions together with the development team. If the amount of technical debt is manageable, one or two sprints might be enough to remove it. If that's not the case, then you may have to allocate more time. Take the Mac OS release *Snow Leopard*, which was made available in 2009 after nearly two years of work. While Snow Leopard didn't provide new functionality, it laid the foundation for future releases, partly by refactoring the code and by improving performance and reliability.

I am not suggesting that you should necessarily spend two years on removing spaghetti code and cleaning up the software. But when faced with a larger amount of technical debt, consider making its removal a product goal on the product roadmap. This will give you the time required to enhance the software quality and future-proof your product.

Set Realistic Targets

Adopt realistic targets against which you can measure performance. For instance, if you use customer satisfaction as a KPI, then you should consider setting a target score, say, 70-85%.[65] While this target might be wrong, it should be good enough to serve as a starting point. If it turns out that it is too low or too high, then adjust it. Validating the product strategy *before* you select any key performance indicators should provide you with the necessary knowledge to set targets that are roughly right.

Additionally, use ranges and ratios to formulate targets like I did for the sample customer satisfaction score (Croll and Yoskovitz 2013). This technique is especially helpful when a significant amount of uncertainty is present, for example, when you

manage a new product, or you undertake a life cycle extension. Using ranges and ratios gives you some leeway, and it avoids the impression that you can set accurate and precise targets for your product.

Combine Quantitative and Qualitative KPIs

As their name suggests, quantitative indicators, such as daily active users or revenue, measure the quantity of something. This has the benefit of collecting "hard" and statistically representative data. Contrast this with qualitative KPIs, such as user feedback. These indicators help you understand *why* something has happened—for instance, why users aren't as satisfied with the product as you expected. Additionally, they help you learn more about the individuals and develop a better understanding of their needs.

Combining the two types gives you a balanced outlook on how your product is doing. If you only use quantitative indicators, you risk losing sight of the most important success factor: the people behind the numbers—the individuals who use and buy your product. If you only apply qualitative metrics, then you are in danger of acting on data that might not be representative of the entire target group.

Take Advantage of Trends

Put the data you collect in context and compare it to other periods. For example, compare last month's earnings to the revenue figures from the previous six months. This helps you spot trends. For instance, if revenue is increasing, staying flat, or declining. Trends allow you to better understand what's happening and to take the right actions. If a decline in revenue is a one-off occurrence, for example, there is no reason to be overly worried. But if it becomes a trend, then you should investigate how you can stop and reverse it—unless you are about to retire your product.

Leverage Lagging and Leading Indicators

Lagging indicators, such as monthly recurring revenue and cost, are backward-focused and tell you about the outcome of past actions. For example, you might use the revenue figures of the last six months to forecast future earnings. If the figures show an upward trend, chances are that this development will continue. While this is a reasonable assumption, a continued revenue increase is by no means certain.

Leading indicators, in contrast, make it easier to understand how likely it is that your product will meet a future goal. Take product quality as an instance. If the code is becoming increasingly complex or buggy, then adding new features will be more expensive, and meeting a profit target will become harder. On the downside, qualitative indicators can be more difficult to apply, and the relevant data can be more challenging to collect.

I therefore recommend that you use both quantitative and qualitative KPIs to understand how well your product is doing and if you can meet the user, business, and product goals.

Regularly Review and Adapt the KPIs

Don't forget to review your indicators on a regular basis: Whenever user, business, or product goals change, your KPIs are likely to change too. New ones may have to be added, and existing ones may have to be removed. For example, if the current product goal is to acquire more users, then market share would be a useful indicator. But if your focus switches to revenue generation, you may add monthly recurring revenue to your KPIs and remove market share, at least for the time being.

Additionally, some indicators are only applicable at certain life cycle stages. Say your product has achieved product-market fit. You may then start measuring profitability and introduce net profit as a new indicator—if the product directly generates revenue. Another example would be a product entering the maturity stage. A metric like referral rate may then no longer be required and might get dropped.

Avoid These Common KPI Mistakes

There are four common pitfalls I see people make when working with key performance indicators. These are: Using vanity metrics; measuring everything that can be measured; blindly trusting a "standard" set of indicators; and allowing senior stakeholders to dictate KPIs. Let's look at these mistakes in more detail and explore how you can avoid them.

First, stay away from vanity metrics, which are measures that make your product look good but don't add any value (Ries 2009a). Say that I've chosen the number of downloads as a KPI for my healthy-eating app. While a fair number of people might download the product, the indicator does not tell me how many individuals actually use it. Instead, it indicates the effectiveness of my marketing efforts. Rather than measuring downloads, I would be better off choosing a metric like daily active users.

Second, don't measure everything that can be measured even if an analytics tool automatically collects the data for you. Otherwise, you risk wasting time by analysing information that provides little or no value. In the worst case, you act on irrelevant data and make the wrong decisions. Think of driving a car. A small number of indicators are helpful for making the right decisions whilst driving, including speed and battery level, assuming you drive an electric vehicle. If the dashboard always showed additional data, such as tyre pressure or ABS status, it would be harder to take in the relevant information. In the worst case, you would overlook important data and, for instance, run out of battery or fuel.

Third, don't make the mistake of adopting a set of "standard" or "must-have" indicators without carefully checking if each KPI is relevant. For example, before you use commonly recommended metrics like customer acquisition cost (CAC), churn, and number of active users for a SaaS product, check that they really help you understand if the product is meeting its goals. If they don't, ignore them.

Fourth, don't allow senior stakeholders to dictate KPIs. While it's great to actively listen to their suggestions and empathise with the individuals, don't make the mistake of including a metric to appease someone. If you believe that declining a KPI is not an option, then you may not be sufficiently empowered. In this case refer to my tips for overcoming a lack of empowerment in the chapter *Introduction*.

A List of Sample KPIs

To conclude the discussion on key performance indicators, let's look at some sample KPIs. The measurements in table 4 are grouped into four sets: financial, customer, product and process, and people. The groups are inspired by David Norton and Robert Kaplan's work on balanced scorecards.[66] Please note that the list of indicators is not intended to be complete. Use the sample KPIs as a starting point and choose only those that are relevant for your product.

Table 4: Sample Key Performance Indicators

Group	Sample KPI	Brief Description
Financial	Revenue	How much revenue is your product generating?
	Cost	What is the cost of developing and launching new major releases or product versions?
	Cost of acquisition	How much does it cost to acquire a customer?
	Profit	How much profit is the product making? Note that you may want to track different profit types, including net and gross profit.
	Customer lifetime value	How much profit do individual customers create across the entire future relationship?
	Cash flow	Is the cash flow positive or negative? If it is negative, when do you expect to reach the break-even point?
Customer	Market share	How big is your market share compared to the competition?
	Adoption rate	Is your product gaining traction in the marketplace? If so, how quickly?
	Engagement	How engaged are the users? For example, how many active daily users does the product have?
	Retention	How many customers are coming back or are renewing their subscriptions?
	Customer satisfaction	How satisfied are the users and customers with the product?
	Cancellation rate	How many contracts are cancelled in a given period?
	Complaints and support requests	How many customer complaints and support queries do you receive? How severe are they?
	Conversion rate	How well are inquiries and evaluations translated into sales?
	Customer and user feedback	Do the users and customers have a positive, neutral, or negative attitude towards your product? What reviews are they providing and what feedback do they share?
Product and Process	User interaction	What are the most and least common user journeys? Where do most drop-offs occur? What are the most and least used features?
	Product quality	Is it easy to change and extend the product? How high are the code complexity and the refactoring potential? What is the test coverage like? How many bugs are found and closed? How severe are they?
	Schedule variances	Are major releases and new product versions deployed on time and on budget and are their product goals met?
	Sustainable pace[67]	How well does the team observe sustainable pace? Do team members regularly work overtime? How high are absence and turnover rates?
	Team knowledge and skills	

People		Does the team have the necessary knowledge to do a good job? Do the team members regularly improve their skills and acquire new knowledge?
	Stakeholder engagement	Do the stakeholders regularly participate in strategy and roadmap reviews and in sprint review meetings?
	Management sponsorship	Do you have the right management sponsor? Does the sponsor show sufficient interest in the product?

Regularly Review and Update the Product Strategy

While the product strategy is key to creating a successful product, it would be a mistake to blindly execute it and assume it will always be valid. As your product develops and grows, and as the market and the technologies evolve, the product strategy must change. You should therefore take the necessary time to regularly review and adjust it.

Five Factors

There are five factors that help you review the product strategy and assess if it is still valid:

1. *Performance*: What do the key performance indicators tell you about the value the product is creating? Does the data show positive, flat, or negative trends? What conclusions can you draw from the analysis? How can you increase the product performance? Are the indicators you are using still relevant, or should they be changed?
2. *Trends*: Are there any new technology, regulatory, or social developments that will affect your product? Do they offer an opportunity to innovate, for instance, to add, remove, or enhance features?
3. *Competition*: Are your competitors launching new products or features? Are there new market entrants? Is your product still sufficiently differentiated and does it still stand out from competing offerings?
4. *Product roadmap*: Has the product roadmap changed? Do the changes indicate that the current product strategy has to be adapted?
5. *Business and product portfolio changes*: Are there any business developments that affect the product strategy? For example, has the business strategy changed or have key people left? Has the product portfolio changed and if that's the case, do these changes necessitate any adjustments to the product strategy?

Review Frequency

To take advantage of opportunities and counteract threats at an early stage, I recommend that you employ the following dual approach, which I also discuss in the *Introduction*:

1. *Continuous reviews*: Collect and evaluate new product performance data and look at any new trends and changes related to your competitors *at least once per week*. Continuously reviewing the product strategy is particularly important when

your product faces a significant amount of change and uncertainty. This is true for brand-new and young products, products that have recently experienced a life cycle extension, and products that serve a dynamic market. In these cases, the product strategy is likely to be volatile and prone to change. It may therefore require frequent adjustments. To carry out continuous strategy reviews, you might want to allocate an hour every day or half a day per week, as I recommend in the *Introduction*.

2. *Collaborative quarterly reviews*: Collaborative workshops are not only great to create a product strategy but also to review and update it. Invite the key stakeholders and development team representatives to the meetings. These should be the same individuals who helped you come up with the current product strategy, as this creates continuity and avoids costly handoffs. To ensure that the product strategy reviews are effective, follow the guidelines I shared in the section *Collaborative Strategy Workshop* in the chapter *Product Strategy Foundations*. These include involving a dedicated facilitator and selecting a clear decision rule, such as consent.[68]

The first of the two measures above helps you avoid nasty surprises such as a competitor leapfrogging you. It increases the chances that you notice early warning signs like declining customer satisfaction so you can act as soon as possible. The quarterly reviews help you consider longer time frames and bigger trends. Involving key stakeholders and dev team members allows you to leverage their collective knowledge, create alignment, and secure buy-in. While it is useful to schedule the reviews in advance, don't wait for the next meeting if there are developments that need to be urgently addressed together with the stakeholders and team members. Instead, hold a collaborative review as soon as possible.

Don't forget to block the necessary time in your calendar, and don't allow urgent issues like sales and support requests to take over and cause you to neglect the strategy work. Otherwise, you are in danger of overlooking opportunities and threats, which will result in more unplanned work in the future.

Four Choices

Once you've reviewed the product strategy, decide what to do. There are four main choices that you have:

1. *No change*: Leave the strategy as it is. This means that the product strategy is still valid; the product is performing well; there are no new trends, no new competitors, and no business strategy, product portfolio, and product roadmap changes that you need to respond to.
2. *Small change*: Carry out small adjustments to improve product performance or respond to a market development. This includes adapting the value proposition and target group, improving the product's standout features, and refining the business goals.

3. *Big change*: Make a substantial strategy change like pivoting, unbundling one of more features, or taking your product to a new market. Such a change may be required when your product is no longer effectively differentiated, when you want to extend its life cycle, when you are facing a disruption in the marketplace, and when the business or portfolio strategy changes. It usually involves creating and validating a new product strategy.
4. *Kill*: Retire the product. This is advisable when you don't stand a realistic chance to achieve product success, when the product has been in decline and the return on investment is no longer attractive, or, in the case of a supporting product, when the offering it supports is being retired. While killing your product may sound drastic, it frees up resources and avoids investing time, money, and energy in a product that is not going to be successful. What's more, despite your best efforts and practising continuous strategizing, there is no guarantee that your product will become or continue to be successful. Innovation is risky, and failure is part of the game.

Combined Strategy and Roadmap Reviews

While using a product strategy and a separate product roadmap is generally beneficial, the two plans are closely connected: The strategy states the approach to attain the vision and achieve product success; the roadmap describes how the strategy will be implemented and which specific benefits the product will create in the coming months. I therefore recommend that you combine product strategy and product roadmap reviews. This saves time and ensures that the two plans are kept in sync: Strategy changes are immediately applied to the roadmap and vice versa.

I suggest allocating two to three hours for a collaborative, quarterly product strategy and roadmap workshop, assuming that it is well prepared and facilitated. This includes having all relevant data and materials available and a skilled facilitator present.

Encouraging the Right People to Attend Strategy Workshops

Sometimes, stakeholders and development team members can be hesitant to participate in strategy workshops. If that's the case, find out why that is. Maybe the individual does not fully understand why their attendance is required; maybe the person had bad experiences contributing to past decisions; maybe they are pressed for time; or maybe there is a conflict with another attendee. Once you understand the reason, ask how you can make it easier for the individual to attend the workshops, for example, by timeboxing the meetings.

If the opposite is the case, and you find that you can hardly make a strategic product decision without a small crowd being present, then there are likely to be different causes at play. Maybe some of the individuals do not trust you to consider their concerns and needs; or maybe you are not fully empowered to make the necessary decisions. Once you've identified the root cause, consider how you can best resolve it, as it tends to be challenging to run effective strategy workshops with significantly more than ten attendees.[69]

PRODUCT ROADMAP FOUNDATIONS

Build a strong foundation and you can reach even the most unthinkable heights.
M.J. Moores

The product roadmap is a great product management tool. But it can cause significant issues if it is not used in the right way. This chapter lays the foundations for an effective roadmapping practice. It introduces important roadmapping concepts and techniques, and it helps you avoid making common roadmapping mistakes.

Understand the Benefits a Product Roadmap Can Offer

As I am writing this chapter, it has turned wintery here in the UK where I live. It's the time of year when I long for some sun and I start thinking about the next summer holiday. Say that my family and I want to spend our vacation in the south of France, and let's also assume that we have decided to drive there from our home in the UK. Then that's a great starting point. But it's not enough—we still have to determine the journey details. This includes deciding if we take the shuttle or a ferry to get across the channel, if we drive via Paris or Lyon, and if we should book a hotel to break up the journey. To make these decisions, I would use a roadmap and determine the best route to get to our destination.

What's true for a road trip also applies to your product. Having a shared vision and a valid strategy is necessary, but it is not sufficient to achieve product success. You also need to describe the journey you want to take your product on and come up with an actionable and realistic plan that states how the strategy is to be executed. In other words, you need a product roadmap.

A product roadmap essentially describes how a product is likely to evolve over time by stating the specific outcomes the offering will create. When used effectively, a roadmap offers the following five benefits:

- It provides a continuity of purpose beyond the next few sprints or the next major release. A common time frame for a roadmap of a digital product is twelve months.
- It states how the product strategy will be implemented thereby bridging the gap between strategy and execution.
- It aligns the stakeholders and development teams so that everyone moves in the same direction.
- It directs and unburdens the product backlog, as I discuss in more detail later in this chapter.

- Finally, it helps you acquire a development budget.

As your product matures and the product strategy stabilises, you may even find that the product roadmap becomes your primary tool for capturing strategic decisions.

But aren't product roadmaps anti-agile?

A product roadmap is simply a strategic product plan that describes how your product is likely to grow, based on your current knowledge. It is neither agile nor anti-agile. To effectively use product roadmaps in an agile context, apply the following three recommendations: First, stop using feature-based roadmaps and adopt goal-oriented, outcome-based plans that don't predict when a feature will be available but describe the specific benefits a product should create, as I describe in the next section. Second, involve the key stakeholders and development team members in creating and updating the plan. This gives people a say, encourages shared ownership, and makes it more likely to create a realistic plan that observes sustainable pace. Finally, get the relationship between the product roadmap and the product backlog right and keep the two plans in sync. (I'll say more about collaborative roadmapping and connecting the roadmap and backlog later in this chapter.)

Take Advantage of Goal-oriented, Outcome-based Product Roadmaps

Traditionally, a product roadmap is an output-focussed plan that maps features like registration, search, and reporting onto a timeline.[70] Such a roadmap essentially states when a piece of functionality will be delivered. This can be reassuring for customers and stakeholders. But it has the following three drawbacks: First, a feature-based roadmap makes it hard to secure agreement, as stakeholders often compete to get their feature on the roadmap. Second, it overlaps and competes with the product backlog, especially when fine-grained features are used. This makes the product roadmap harder to understand, and it increases the effort to keep it up-to-date. Third, the features are sometimes regarded as a commitment rather than a part of a high-level plan that is likely to change. This limits your ability to experiment and learn, to discover the best way to address the user and customer needs and create value for the business.

These drawbacks are largely avoided by using a different product roadmap type: a *goal-oriented, outcome-based roadmap*, which is sometimes called *benefits*-based or *themed* roadmap. As its name suggests, this roadmap focuses on product goals or outcomes, such as acquiring customers, increasing engagement, and future-proofing the product by removing technical debt. Features might still be used, but they are dependent on the goals: Every feature must serve a goal and be required to create a specific outcome. Figure 23 illustrates the two different roadmap types.

When applied correctly, a goal-oriented product roadmap can offer the following six benefits: First, it communicates why it is worthwhile to progress the product by stating the specific benefits a product is likely to create. Second, it improves stakeholder alignment by establishing shared goals that direct the work of the individuals and give them the autonomy they need to do a good job. Third, it gives the development team the direction required to do a great job without fixing the detailed functionality upfront. Fourth, it neatly connects the product roadmap and the product backlog by using the next roadmap goal to focus and direct the product backlog. Fifth, goals make the roadmap less

volatile: They tend to be less susceptible to change compared to feature lists.[71] Lastly, goal-oriented roadmaps are compatible with the goal-setting method objectives and key results (OKRs): You can think of the goals as objectives and the other roadmap elements as the key results.

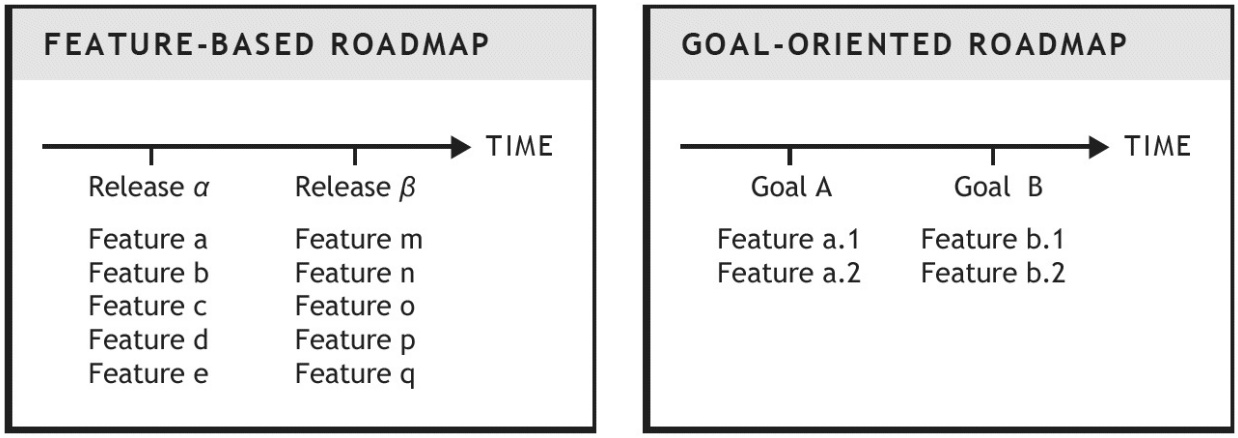


Figure 23: Feature-based vs. Goal-oriented Product Roadmap

These benefits make goal-oriented, outcome-based product roadmaps preferable over traditional, feature- and output-focused plans, especially for digital products that exhibit uncertainty and change virtually across their entire life cycles.

Which roadmapping tool should I use?

When it comes to choosing a roadmapping tool, my advice is to opt for one that is easy to use and that allows the stakeholders and dev team members to view and contribute to the plan. If you are unsure, then start with a simple spreadsheet or electronic board before you decide which (if any) specialised product roadmapping tool is right for you. The biggest mistake you can make is to select a powerful tool and hope that it will fix your roadmapping challenges. But as Grady Booch, one of the creators of the Unified Modelling Language (UML) once said, "A fool with a tool is still a fool."

Practise Collaborative Product Roadmapping

No matter how well thought-out your product roadmap is, it is worthless if the key stakeholders and the development team members—also known as *players*—don't understand and support it.[72] To ensure the roadmap is shared involve the players in the roadmapping decisions, preferably in the form of a collaborative workshop, as I explain in more detail in the next chapter. Ideally, the same individuals who helped you create and validate the product strategy also create and update the roadmap. This takes advantage of the trustful connections that have hopefully been established and it avoids loss of knowledge and handoffs.

But don't forget that you, the person in charge of the product, should lead the roadmapping work and guide the creation of a new or updated plan. While I'd like to

encourage you to embrace a collaborative mindset, attentively listen to the ideas and concerns of the players, and empathise with them, you should not allow any individuals to tell you what to do or to dictate goals or features. Otherwise, your roadmap is likely to be a weak compromise rather than a compelling plan that results in a successful product. Collaborative roadmapping does not mean that everybody gets their way or is necessarily super happy with every single decision. It means leveraging people's expertise to create a product roadmap that maximises the value the product creates and that attracts as much support as possible. Therefore, have the courage to decline stakeholder requests if they are not in line with the product strategy and make a decision if no agreement can be reached.

Base Your Roadmap on a Validated Product Strategy

For a product roadmap to be effective, it must be actionable and realistic. The best way to achieve this is to base the plan on a validated product strategy, a strategy whose key assumptions have been tested and which does not contain any significant risks.

Take the road trip example I shared earlier. There is no point in choosing a specific route and booking a hotel to break up the journey if I am not sure that driving to southern France is the right approach. You should therefore ensure that you create and validate an overarching product strategy, as described in the preceding chapters of this book, before you develop a product roadmap. Otherwise, your plan may turn out to be unrealistic. This might cause the stakeholders and dev teams to lose trust in the roadmap and possibly also in your ability to guide them.

Following this approach provides you with another benefit: It allows you to systematically connect your product roadmap with the product strategy. To achieve this, consider the needs and business goals in your strategy and use them to discover the right product goals. To put it differently, every goal on your roadmap should help move you closer towards meeting the needs and business goals. This way, the product strategy directs the roadmap and helps determine its contents, as Figure 24 shows.

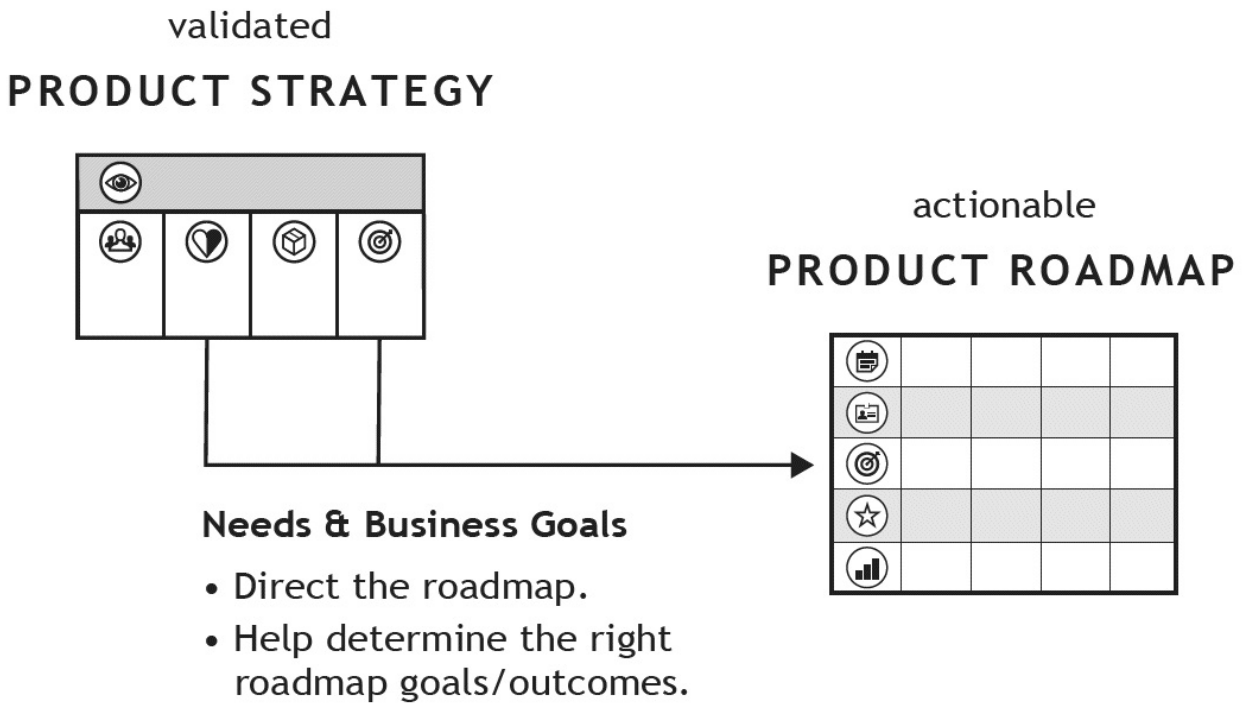


Figure 24: The Product Strategy Directs the Product Roadmap

Using the product strategy to guide the product roadmap ensures that the two plans are consistent: The roadmap details the strategy and states how it is likely to be implemented. Bear in mind, though, that bigger roadmap changes may trigger product strategy updates. Say that you miss achieving several consecutive goals on the product roadmap. This might indicate that you cannot meet the needs and business goals stated in the product strategy. Consequently, you might have to adapt them. The relation between the strategy and roadmap is therefore bidirectional: The strategy directs the roadmap, and the roadmap influences the strategy.

Get the Relationship Between the Roadmap and the Product Backlog Right

The product roadmap and the product backlog are two important product management plans. Each has its own strengths and weaknesses. While the product roadmap is a strategic plan, which describes how your product is likely to develop across several months, the product backlog is tactical in nature. It contains the details necessary to progress the product and directs the work of the development teams. The backlog includes epics, user stories, non-functional requirements, design sketches, and other artefacts that describe what the product should look like and do. Both plans nicely complement each other when applied properly, as Figure 25 shows.[73]

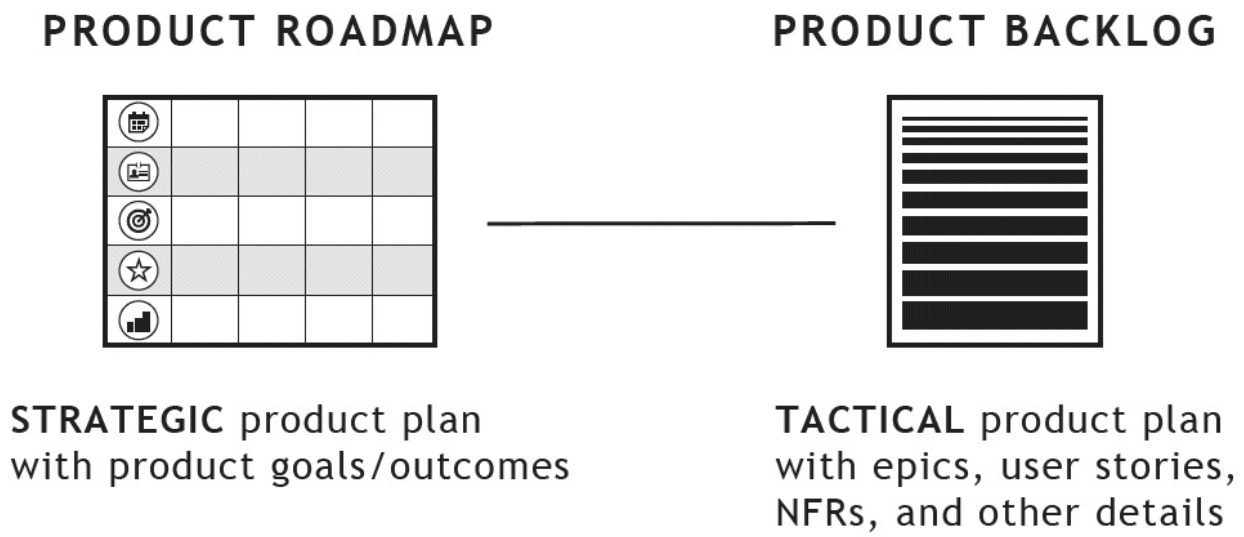


Figure 25: Product Roadmap and Product Backlog

Unfortunately, I find that product roadmaps sometimes contain too many details, including epics and user stories, and that some product backlogs look too far into the future. This blurs the line between the two plans, and it results in a roadmap that is difficult to understand, prone to change, and hard to manage. You should therefore keep the two plans separate and leverage their respective strengths. Employ the roadmap to describe your product's overall journey and the backlog to capture the details. Adopting this method has helped one of my clients, a major games studio, reduce the size of its product backlog from over a thousand items to less than one hundred.

Distinguish Internal and Public Product Roadmaps

To put the right contents on a product roadmap, it is helpful to differentiate between internal and external, public roadmaps. As its name suggests, an internal product roadmap—which I focus on in this book—is visible only within the company that creates and provides the product. It helps align the stakeholders and dev teams and guides the work of the individuals, such as creating the necessary marketing collateral, preparing the sales channels, and making the right development decisions. External product roadmaps, in contrast, are visible to the users and customers. Such a roadmap is sometimes used as a marketing tool—to show that the company is committed to the product and has great ideas of how to improve it in the future.

Some organisations work with separate internal and external roadmaps, whereas others use just one plan. I recommend the first option: Use two plans if you want to show an external product roadmap. Additionally, derive the external plan from the internal one. Reduce the level of detail shown and remove any dates. This avoids setting customer expectations that you might not be able to meet.

Avoid These Common Roadmapping Mistakes

While the product roadmap can be tremendously helpful, I find that it is not always effectively used. There are six common mistakes, which you should avoid:

Stakeholders Determine the Roadmap Contents

I find it not uncommon that stakeholders want to get specific features onto the product roadmap. While it is important that you are open to their views and concerns, you should not allow stakeholders to dictate the roadmap contents. Your job is not to please the stakeholders, but to achieve product success. You should therefore decline stakeholder requests if they aren't aligned with the product strategy and if they don't help maximise the value the product creates.

The Roadmap is Seen as a Fixed Plan

Some people view the product roadmap as a plan that is set in stone. But any roadmap is based on what you currently know. While it's good to have confidence in your roadmap, it will change as you start implementing it and learn more about how to best meet the user, customer, and business needs. This, in turn, is a good thing: It helps you maximise the value the product creates and offer the best possible product—instead of blindly executing a plan that might be outdated. You should therefore review and adapt the roadmap on a regular basis, as I discuss in more detail in the chapter *Product Roadmap Reviews*.

The Roadmap is Speculative

As useful as a product roadmap can be, there is no point in creating a speculative plan that is built on wishful thinking. It would misguide and disappoint the stakeholders and development teams who use it. Therefore, do not create a roadmap if you haven't got a validated product strategy in place or if you cannot see beyond the next product goal—which can happen especially when you work on a brand-new, innovative product.

In the first case, delay using a roadmap until you have created and validated a product strategy. Then use the strategy to derive the right roadmap. In the second case, only use the product goal you've identified for now. As you work towards the goal, you will hopefully understand better how you can progress your product in the future and be able to create a realistic product roadmap.

The Roadmap Results in a Death March

A product roadmap that is overambitious and contains unrealistic goals can turn the development effort into a "death march" where the development team members regularly work overtime, are continuously stressed, and end up being exhausted. Consequently, creativity, motivation, and productivity drop; people's wellbeing and health suffer; and software quality is often compromised making it harder to update the product in the future. You should therefore ensure that your product roadmap is realistic and that it supports sustainable pace so that the team members can implement it without being overworked, losing motivation, and falling ill.

The best way to develop a realistic product roadmap is to involve the development team members in the work. Listen to their views with an open mind, and don't pressurise them to agree to the roadmap contents. Instead, take their concerns seriously and iterate over the plan until it is feasible using the techniques I share in the next chapter. Additionally, regularly review and adapt the product roadmap to ensure that it stays realistic.

The Roadmap Contains Epics and User Stories

Epics and user stories describe end-user functionality in the form of a narrative. A user story is usually small enough to fit into a sprint; an epic is a bigger story that describes a larger piece of functionality. While epics and user stories are very popular and commonly used by agile teams, it would be a mistake to include them in your product roadmap. This would make the plan too detailed, create an overlap with the product backlog, cause the roadmap to be more prone to change, and make it harder to understand how you want to progress your product. You should therefore use epics and user stories in the product backlog, but not on the product roadmap. Use the roadmap as a strategic plan that describes the desired outcomes and the backlog as a tactical tool that captures the details required to achieve them.

The Roadmap is Mistaken for a Release Plan

A release plan forecasts how a major release or a new product version is developed.[74] You can think of it as the agile equivalent of a project plan. Release plans come in different shapes and sizes depending on the process model used. In Scrum, the release burndown chart is the default release plan. This chart helps you track the progress from sprint to sprint and anticipate if a product goal can be met on time and budget—or how long it will take and how much it will cost to do so. This allows you to guide the work of the development team and make the necessary adjustments, such as, removing functionality from the product backlog. In other words, a release plan helps you maximise the chances of meeting a product goal.

The product roadmap, however, states how a product is likely to evolve over a longer time frame, say, the next 12 months. It is not based on the product backlog but on the product strategy, and it contains several product goals. Therefore, don't confuse the two plans but clearly distinguish them and use separate artefacts to capture them. I say more about release plans in the last chapter of this book.

PRODUCT ROADMAP DEVELOPMENT

Good fortune is what happens when opportunity meets with planning.
Thomas Edison

This chapter describes practices to help you get your product roadmap right—to create a realistic and actionable plan that contains the right goals and outcomes. I also discuss techniques to prioritise the roadmap goals, determine dates and cost, and derive a focused product backlog. But let's first look at the steps that you should take to develop an effective goal-oriented product roadmap.

Take the Right Steps

When you plan a road trip, there are certain points you usually need to consider like the route you want to travel on and the stops you'll have to make to recharge or refuel the car. Similarly, when you create a goal-oriented roadmap, there are eight steps that you should take. I list these steps below and I describe them in more detail in the remainder of this chapter. The steps are:

1. Identify the product goals—the outcomes you want to achieve with your product.
2. Make the goals specific and, if possible, measurable.
3. Prioritise the goals.
4. Capture selected coarse-grained features that are necessary to meet the goals.
5. Determine realistic, achievable dates or time frames.
6. Estimate cost (unless the budget is fixed).
7. Iterate over the plan, trade-off goal attainment, on-time delivery, and budget adherence; adjust the roadmap until it is actionable and realistic.
8. Derive the product backlog from the product roadmap so that you can start working on the first product goal.

I recommend that you take the steps in the order stated—unless the dates or time frames are fixed, and your product offers new benefits for instance, every three months. If that's the case, start with setting the dates. Next, decide which product goals you can realistically achieve without sacrificing sustainable pace and product quality. Then continue with step two from the list above, but skip the fifth step, as you've already determined the dates.

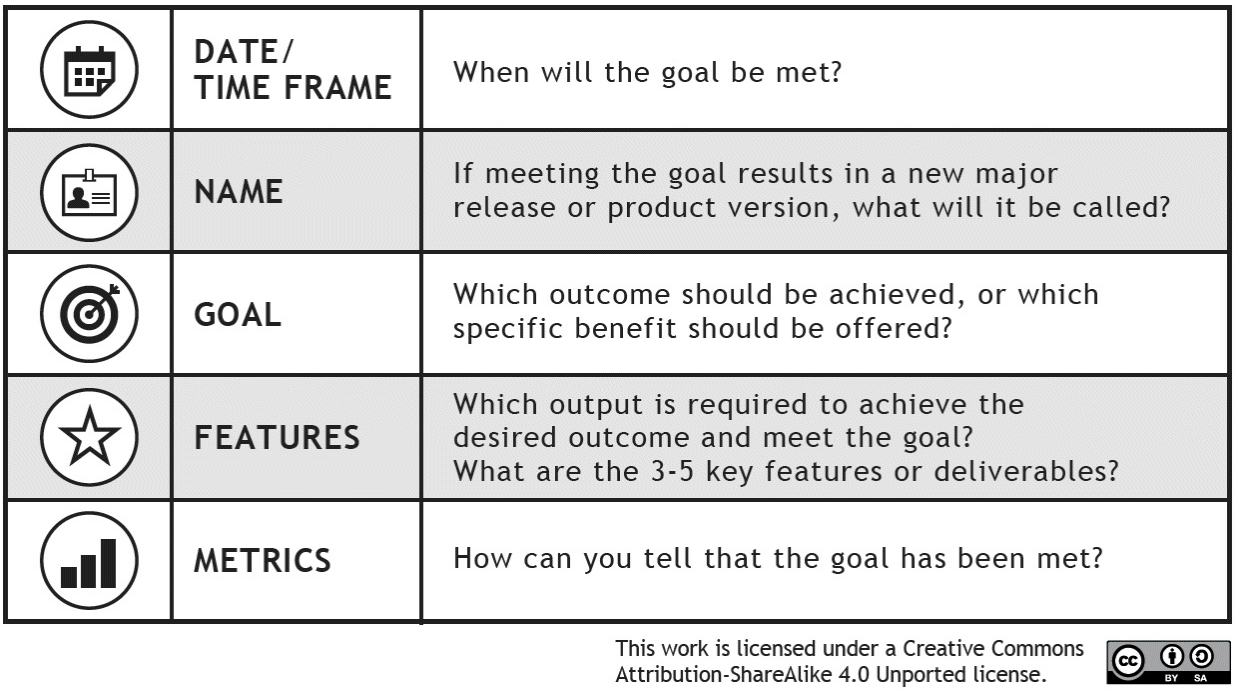
While the steps above might sound like a lot of work, in practice it usually isn't that bad. Goal-oriented roadmaps often contain between three and six product goals, depending on the time frame covered and the size of the goals. Say that you use a twelve-

months roadmap with quarterly goals. You will then have to select four product goals, prioritise them, identify three to five coarse-grained features per goal, and determine dates and cost. This shouldn't take longer than a few hours, assuming that you have done the necessary strategy validation work.

Make sure that you involve the key stakeholders and development team members in the steps above, preferably by using a collaborative workshop, as I explain in more detail later in this chapter. This allows you to leverage the expertise of the individuals, ensure that the plan is realistic and clear, and increase the chances that people follow the roadmap and put it into action.

Capture Your Roadmap with the GO Template

To help you take advantage of goals and outcomes on your product roadmap, I have created a template, which is called the *GO product roadmap*. Figure 26 shows this template.



	DATE/ TIME FRAME	When will the goal be met?
	NAME	If meeting the goal results in a new major release or product version, what will it be called?
	GOAL	Which outcome should be achieved, or which specific benefit should be offered?
	FEATURES	Which output is required to achieve the desired outcome and meet the goal? What are the 3-5 key features or deliverables?
	METRICS	How can you tell that the goal has been met?

Figure 26: The GO Product Roadmap Template

Let's look at the five rows of the GO product roadmap in Figure 26. The first row captures the *date* or the *time frame* when a goal should be met—for example, 1 September 2022 or third quarter of 2022.

The second row gives you the option to state a *name*. This is useful when meeting the product goal results in a new major release or product version. Think of iOS 15 or Windows 11, for instance.

The third row is the most important one: It states the specific *product goal* you want to achieve, the benefit the product should provide and the outcome it should achieve. An effective product goal describes the purpose for progressing the product, and it is in line with the needs and business goals stated in the product strategy. Sample product goals include acquiring and retaining users, increasing engagement, reducing technical debt, and improving conversation.

The fourth row lists the product's *features*. These are the outputs that are required to meet the goal. Always ensure that any feature stated is truly necessary to achieve the desired outcome, keep the features big and coarse-grained, and don't use more than three to five features per goal, with a preference for the smaller number. It's a common mistake to add too many features to a roadmap: This makes the plan too detailed and susceptible to change, and it creates an overlap with the product backlog. Remember that a goal-oriented roadmap primarily communicates the value your product should create. The product details, such as epics and user stories, should be in your product backlog, not on the roadmap.

The fifth and final row captures the *metrics* to determine if a product goal has been met—for example, x number of users employ the product for at least thirty minutes per day within two weeks after the software is released. Stating the metrics ensures that the goals on your roadmap are specific and measurable.

If the GO roadmap template resonates with you, then you can download it from my website, romanpichler.com. Note that its licence agreement allows you to tailor it to your specific needs. You might add a new row that captures the marketing and sales channels, for instance. You might also include cost targets for the goals, or you might remove the name row altogether if you don't use any major releases.

Set the Right Product Goals

At the heart of any goal-oriented, outcome-based product roadmap are its goals. But finding the right goals is not always easy. How can you tell which goals you should select? This section introduces two methods for choosing the right product goals: deriving them directly from the needs and business goals in the product strategy and determining them with the help of the key performance indicators (KPIs).

Derive the Product Goals from the Needs and Business Goals

To discuss how the needs and business goals stated in the product strategy help you select the right products goals, let's take the healthy-eating app as an example, which I introduced earlier in this book. Say that its needs statement is to "reduce the risk of developing type-2 diabetes," and the business goal is to "create a new revenue source." Let's also assume that the business model I have chosen is freemium, giving away a free basic version and generating revenue through in-app purchases.

With this information in place, I can ask myself, what a first, concrete step is to meet the needs and the business goals. My answer might be, "help users understand their eating habits and acquire an initial user base." What I have done here is breaking down the two

higher-level goals into a more specific subgoal. Applying this method does not only help determine the right product goals. It also systematically connects the product roadmap to the product strategy—the former is literally derived from the latter.

Note that I used a *compound goal* in the example to capture the desired outcome. The goal consists of two closely connected parts—a user part, "help the users understand their eating habits," and a business one, "acquire an initial user base." This allows me to clearly describe the specific value the product should create for the users *and* for the business. Additionally, it avoids the risk of neglecting the users' needs by being fixated on business outcomes.

If I can see further than the initial offering or MVP for my new healthy-eating app, I would derive additional product goals. These might be: "Help the users improve their eating habits and grow the user base and help the users get fitter and generate revenue in the form of in-app purchases." Together, these goals form a meaningful narrative. They describe how the product is likely to evolve in the coming months: Each goal helps implement the product strategy and it is a step towards realising the product vision.

Use KPIs to Determine the Product Goals

Deriving product goals directly from the product strategy works well when you look after a product that experiences a significant amount of innovation and change, for example, when the product is brand-new or young and changing. Unfortunately, the method is not effective for products that are stable or mature, that experience incremental changes and smaller updates. Luckily, there is an alternative: using the product's key performance indicators (KPIs) to identify the right roadmap goals.[75]

Say that engagement is a key indicator for my healthy-eating app and that it has been declining for the past three months. I may then want to choose a product goal that addresses the issue and improves the metric. I might do this by enhancing the user experience, adding a new feature, or providing performance and stability improvements, depending on what causes the engagement to be low.

Another example would be an increase in software bugs and code complexity, which indicates that the product health is degrading, and that the software is becoming more difficult to extend and maintain. In this case, I might choose a product goal like "future-proof the product by reducing technical debt" to address the issue and improve the health of your product.

Employ Single Product Goals

I find it helpful to use one product goal at a time instead of concurrently working on multiple goals. This approach offers the following two benefits: First, it creates alignment and focus, as everyone is following the same goal. Second, it makes it easier to track progress—to understand if the goal has been met and if the desired outcome has been achieved. Therefore, avoid setting several product goals for a given period.

Don't Mistake Features for Goals

A product goal should describe the reason for progressing the product and the specific benefit or outcome it should create. But it's not uncommon in my experience that features are mistaken for goals. Let's use my healthy-eating app again to illustrate this mistake.

Say that I want to explore what the product could do for the users, and I come up with "measure calorie intake and determine blood sugar level." Do these statements then qualify as product goals? I don't think so. In my mind, they describe product capabilities and characterise the solution. But they don't state why it is worthwhile to progress the product.

A good test for product goals is therefore to ask the why question. For example, why would it be helpful to measure calorie intake? The answer would then reveal the true goal, such as "help the users improve their eating habits." Therefore, be careful not to mix up goals and features, and ensure that your product goals always capture the desired outcome and not the output.

Can Product Goals be Captured with OKRs?

OKRs, which stands for *objectives and key results*, are a method for setting and tracking goals. An objective describes what is to be achieved. The key results state how you accomplish the objective (Doerr 2018). OKRs were originally invented by Andy Grove at Intel in the 1970ies to set hard, measurable goals. This makes the method well suited to capture product goals, at least in theory. You can view a goal as the objective, and you can regard its date, metrics, and features as the corresponding key results. But as I mentioned in the *Introduction*, I find that using OKRs leads to a text-heavy approach that makes it comparatively hard to create, understand, and evolve a product roadmap. It feels like turning back the clock and not taking advantage of the visual product management tools developed in recent years, including my GO product roadmap.

Make the Goals Measurable

When you go on a road trip and visit a tourist attraction along the way or stop overnight at a hotel, you can tell whether the attraction was interesting, and the hotel was comfortable. In the same way, you should be able to tell if your roadmap has the anticipated impact and if the goals have been successfully met. Taking the following two steps will help you set measurable product goals:

First, iterate over them until they have become specific enough to state a target. It is common in my experience to start with coarse-grained product goals, which are neither specific nor measurable and which consequently should be reworked and improved. Second, select the measurements that will help you determine if a goal has been met and if the desired outcome has been achieved. Then add them to the metrics section on your product roadmap, assuming that you use my GO roadmap or a similar template.

To successfully carry out these two steps, consider *how* you will be able to tell if the goal has been met. For instance, if your goal is to acquire between 5-10% new customers, then determine how you are going to measure if the objective has been achieved. Does a successful acquisition require that an individual registers with your website, for example? Or should you measure if the number of unique visits has increased? And what does "new" mean? Should the customers belong to the same market or market segment that is

currently served, or do you intend to reach out to an additional one? Similarly, if you want to reduce technical debt by 50%, for instance, then ask yourself how you can tell that you will have met the target? Are you going to measure if the code complexity or the refactoring potential has declined, or possibly both? Additionally, state *by when* the goal should be met. In the case of an acquisition goal, you may have to wait several days or even a few weeks after the software is released before enough data has become available so you can understand whether the desired benefit has materialised.

If, however, you find that making all the goals on your roadmap measurable is too difficult at present, then focus on the first product goal, and ensure that at least this goal can be measured. Leave the other goals as they are for now and rework them when you review the product roadmap. If you struggle to make the first goal on your roadmap measurable, then this may indicate that you'll have to carry out more strategy validation work.

Balancing Storytelling and Precision

One of the benefits of using a product roadmap is the ability to describe the journey you want to take your product on and guide the stakeholders and development teams. The more precise and measurable you make the goals on your roadmap, the harder it can be to understand the plan and feel inspired by it. Fortunately, the approach described above balances storytelling and measurability: Using goals that are easy to comprehend and complementing them with metrics that are captured in a separate section on the roadmap allows you to tell a story about the progress you're likely to make without losing the ability to measure if the outcomes are achieved.

Prioritise the Product Goals

Once you've selected the right product goals, you should consider the order in which they will be worked on. In other words, you should prioritise them. In this section, I discuss three techniques to help you with this: using semantic dependencies that exist between the goals, determining the cost of delay, and considering dependencies to other products.

Semantic Dependencies

Exploring the semantic dependencies between product goals can help you sequence them so that each goal forms part of a logical progression and is a further step towards the overall needs and business goals that are stated in the product strategy. Let's explore how this technique can be applied using the three sample goals below, which I've also used in the section *Set the Right Product Goals*. Note that I have used non-measurable statements to make the goals easier to understand.

1. Help the users understand their eating habits and acquire an initial user base.
2. Help the users improve their eating habits and grow the user base.
3. Help the users get fitter and generate revenue in the form of in-app purchases.

I chose the order above, as users first have to understand their eating habits before they can improve them. Improved eating habits are, in turn, the basis for getting fitter and adopting a healthier lifestyle. Similarly, I would first have to acquire a large enough user base before I can generate revenue, assuming that a freemium business model with in-app purchases is used. In other words, the third goal depends on the second one, and the second goal depends on the first one.

Using semantic dependencies between goals helps you tell a convincing story about the likely development of your product. For a brand-new product, like my healthy-eating app, this might mean that you start with user acquisition followed by activation, retention, and finally revenue generation, influenced by the product's underlying business model.

Cost of Delay

Using semantic dependencies between product goals is great when your product experiences change and carries some forward momentum. But when your product enjoys a phase of stability—be it that it is experiencing steady growth or that it has entered the maturity stage, the product goals you set can be seemingly unconnected and lack clear semantic dependencies. Consequently, they can't be easily ordered to form a logical sequence or a proper narrative.

Faced with such a challenge, I recommend using the *cost of delay* to get the roadmap prioritisation right. To put it simply, ask yourself how big the loss or severe the disadvantage is likely to be when you delay each goal. For example, if you are unsure whether you should first enhance the user experience to sustain engagement or fix bugs to prevent churn, then identify the impact of delaying each goal. Once you've determined the cost of postponing the items, address the one with the biggest cost of delay first, then the item with the second biggest cost, and so forth. If you find it hard to quantify the expected benefits, then consider if the value of a product goal is high or low and if meeting it is urgent or not. This results in a low, medium, or high cost of delay, as shown in Figure 27. Tackle the product goal with the highest cost of delay first, then work on the medium ones, and finally on the goals with a low cost of delay.

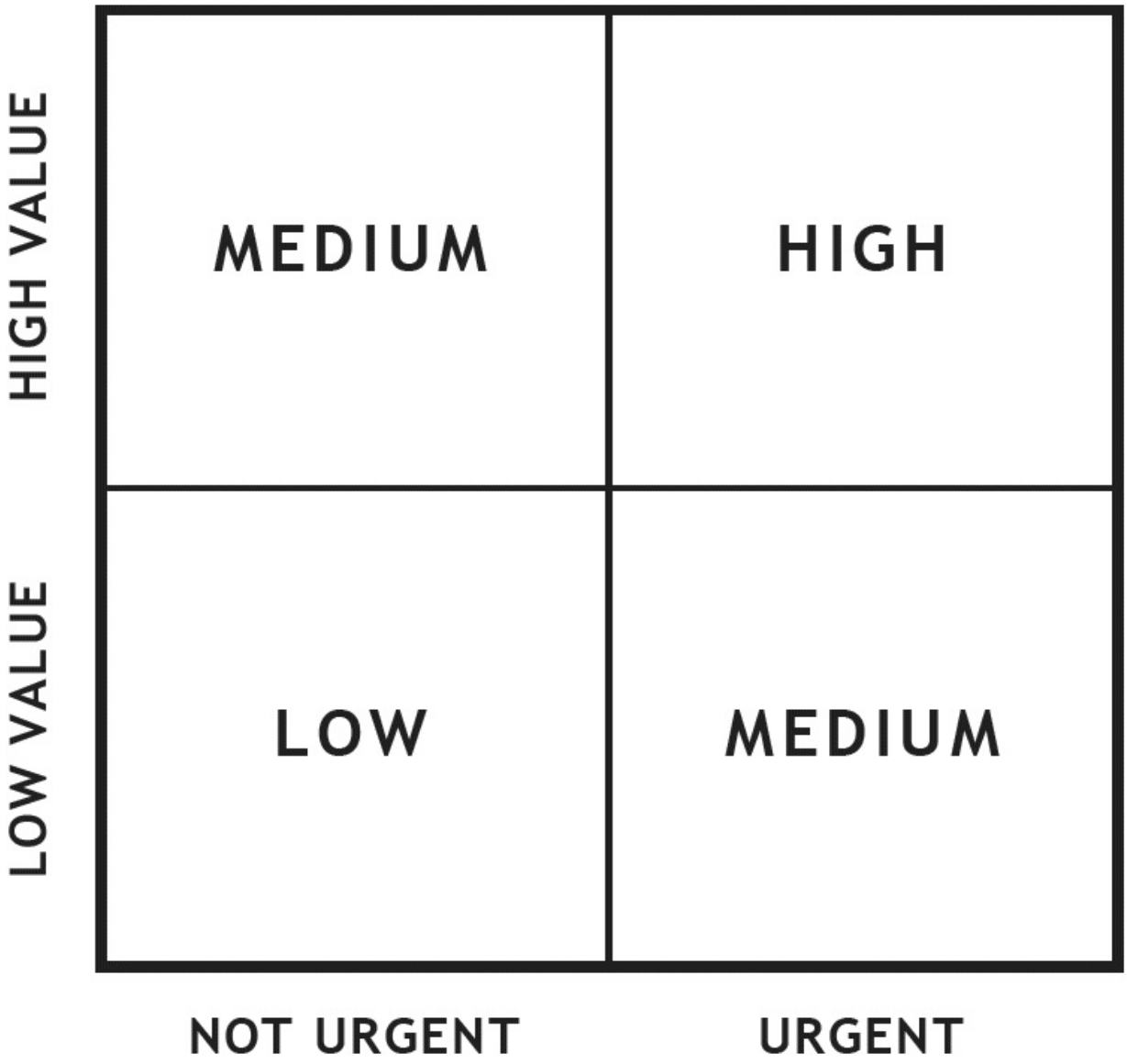


Figure 27: Qualitative Cost of Delay Matrix, based on Arnold (2016)

You can take this approach further and use a related method called *cost of delay divided by duration*, or CD3. As its name suggests, you determine not only the cost of delay for a product goal, but you also establish how long it is likely to take to meet the goal. Dividing the cost of delay by the time estimate gives you a CD3 score. Once you've calculated the score for each product goal, state them in descending order. The goal with the highest value goes first followed by the one with the second highest and then the third highest score. I find this technique particularly helpful when the number of people working on development teams can't be easily changed, and when the durations to meet the goals vary.[76] The method does assume, though, that the development team members can come up with rough but realistic time estimates. If that's not the case, use the *cost of delay* instead of CD3.

Inter-Product Dependencies

Finally, don't forget that your product might be constrained by dependencies to other products, which can influence the prioritisation of the roadmap. This is particularly relevant for supporting products. Take the example of one of my clients, a major games studio. The group I worked with develops one of the physics engines used in the company's computer games. A physics engine is a supporting product that offers complex animations. Without it, the game characters would make slow, robotic movements, and playing the game would not be much fun. Consequently, the release dates and the animation requirements of the games significantly influence the physics engine roadmap and its prioritisation. The same is true for a software platform that encapsulates shared assets and supports end-user facing products. The products built on it influence not only the product goals but also their prioritisation. If you manage a supporting product, you will therefore have to align your product roadmap with the plans of the supported products. A portfolio roadmap can help you with this, as I discuss later in this chapter.

While dependencies are a fact of life, they can be problematic, especially when your product experiences innovation, change, and uncertainty. That's typically the case for new and young products as well as products whose life cycle is being extended. In these cases, a high degree of autonomy and loose coupling to other products are desirable. You can reduce dependencies by changing the product boundaries and the team setup. To apply the first option, you might unbundle the product and turn a feature into a new product; you might bundle smaller products into a larger one; or you might encapsulate shared assets, components, and services in a platform, as described in part one. Bear in mind, though, that all these changes are likely to require some architecture refactoring. To improve the teams and their setup, form dedicated product teams—teams that are organised around products, ensure that each team works on one product, and keep the teams stable. As a rule of thumb, prefer teams that deliver end-to-end functionality and can implement an entire vertical slice over teams that are organised around architecture building blocks like services and components, at least for as long as your product experiences uncertainty and change.

Get the Features on the Roadmap Right

As powerful as product goals are, they are often not enough to understand what needs to be done to create the desired outcomes. This is where features come in. As mentioned previously, I use the term *feature* to describe a product capability—a big piece of functionality. Think of the ability to understand calorie intake or to seamlessly integrate with smartwatches—to use two features of my healthy-eating app as examples. But features have a dark side: If you are not careful, they can take over your roadmap and encourage people to worry more about the output than the outcome.

To avoid this from happening, apply the following three tips: First, always start with the product goals. Then derive the features from them and assign any existing features to the goals they support. After all, features exist to meet a goal and help generate a specific benefit. Second, don't use more than three to five features per goal. Only capture the output that is essential to meet a goal. Remember: The roadmap is a strategic plan that

communicates the outcomes a product will achieve; it should be not a feature list. Third, keep the features coarse-grained. Determine the detailed functionality as part of the development work and capture it in the product backlog. This avoids that the roadmap and backlog overlap, and it helps ensure that the product offers the right functionality, assuming that you run sprints and collect user feedback on early product increments.

If you are faced with a feature request, check if the feature would help meet an existing product goal. If this is the case, consider adding the feature to the roadmap and removing or changing another feature that belongs to the same goal. If no corresponding goal exists, explore if it is worthwhile to change the plan, adjust an existing product goal or introduce a new one so that the feature can be added to the roadmap. A cost-benefit analysis can help you make the right decision. To apply this technique, ask the development team members to estimate the rough effort required to make the change. Then determine the benefit it is likely to create. To better understand the latter, you might consider how many users would benefit from the change and how confident you are that the desired impact can be achieved, as the RICE method suggests.[77]

While it is important, of course, to objectively assess a roadmap change, it is not enough. I find that stakeholders must feel appreciated and understood to be open to a rational conversation and to agree that their feature cannot be added to the plan. I therefore recommend that you attentively listen to the stakeholders who request roadmap changes and empathise with them, even if you don't find the individuals particularly likeable and if you disagree with their requests. Additionally, adopt a collaborative approach. Don't allow stakeholders to push through their individual requests. Instead, discuss them together with the other stakeholders and the development team representatives, for example, in the next roadmap review meeting, which I describe in the next chapter.

Determine Dates

If dates should be shown on a product roadmap, is a question that has attracted much debate in product management. Some people passionately argue that they should be banned from roadmaps, while others assert that they are useful. Here is my view: Whenever you work with a public, customer-facing product roadmap, avoid using dates. Instead, use time frames that are big and vague so you have enough wiggle room. For instance, you might choose six-months time frames, or just talk about what you plan to do now, next, and later.[78] But when you work with an internal roadmap that aligns development teams and stakeholders, I would recommend stating dates or specific time frames.

There are two reasons for this: First, some products must meet specific dates to achieve success. Take, for example, products like computer games and smartphones, whose main sales tend to take place prior to Christmas. Having these products ready on time is essential, as a delay would have a significant revenue impact. Second, even if your product is not affected by any deadline, it is usually helpful to consider dates or time frames when developing a product roadmap. This allows you to understand if the plan is

realistic—if the desired goals can be fully met within an expected period and without sacrificing sustainable pace or quality.

To set dates, people traditionally determine what needs to be done and then work out how long it will take. Consequently, requirements are compiled; work packages, tasks, and dependencies are identified. Finally, tasks are estimated, and a delivery date is set. This approach tends to work if all requirements and the software architecture can be correctly specified upfront, and if no major changes occur during the development effort. But it does not work well when innovation, uncertainty, and change are present.

A better and usually faster way to determine dates on a roadmap is to investigate when a goal must be met to achieve the desired outcome. Two techniques can help you with this, the *window of opportunity* (Wysocki 2013) and a *steady release cadence*.

Window of Opportunity

The window of opportunity describes the time frame during which a goal must be met to realise the desired outcome. This technique is especially helpful when your product is seasonal and when the uncertainty present makes it very difficult, if not impossible, to employ a traditional approach and break features into epics and user stories without second-guessing them and ending up with a product backlog that's big, detailed, and speculative. In the case of seasonal products, the window of opportunity is usually predetermined. Take, for instance, smartphones, which I mentioned earlier as an example. The window of opportunity for phones is the pre-Christmas period, and major manufacturers like Samsung and Apple ensure that new models can be ordered by October.

If your product experiences uncertainty—be it that it is brand-new or that you are making a bigger change to an existing one, then determine how dynamic your market is. Is it changing quickly, or is it comparatively stable? Are new market entrants appearing? Are existing competitors working on a similar product, or are they likely to make comparable changes to an existing offering? If you have done the necessary strategy validation work including some competitive analysis, you should be able to answer these questions and choose the right time frame. Once you have started the development work, track the actual progress towards the product goal. This will help you understand if the time window you came up with is realistic. If that's not the case, adjust the product roadmap.

Steady Release Cadence

Using a steady release cadence means releasing software at a fixed rhythm—for example, every six weeks, two months, or three months. You essentially timebox your product goals and choose the same length for all of them.[79] This provides you with two benefits: First, it simplifies and improves the planning process. As the date is fixed, you can focus on setting the right goals and sizing them so that they fit into the predefined time boxes. Over time, you will learn how much you can actually achieve within a timebox, and you will get better at setting realistic goals. Second, your users and customers will be able to

regularly enjoy product improvements. This can give you a competitive advantage: Competitors will have to match your pace.

A steady cadence with short release cycles can be particularly helpful once your product has stabilised, be it that it is growing steadily or that it has entered the maturity stage. Take, for instance, the Google Chrome browser. It took Google about two years to develop version 1.0, which the company released in December 2008. For the next two years, the Chrome team released a new version about every four to six months. Afterwards, a new version was released every six weeks. By then, the product was well into the growth stage and had become the third biggest browser in terms of usage share.[80]

Estimate Cost Top-Down

There is no free lunch, as the saying goes, and implementing a product roadmap obviously incurs cost. Determining expenditure is useful to understand how much money is likely to be required to develop the product or—if the budget is predetermined and cannot be changed—to find out whether the roadmap can be implemented.

Bottom-up vs Top-down

Traditionally, development cost is determined bottom-up. Based on a comprehensive and detailed requirements specification, work packages and tasks are derived, which are then estimated. This allows you to calculate the delivery date and development cost. But as mentioned before, this approach does not only require that innovation, uncertainty, and change are largely absent. It is also costly. It can take weeks to complete the necessary work.

I therefore recommend that you use a different method. Instead of following a traditional approach—which would require you to derive a product backlog from the roadmap, detail and estimate its items, and forecast the backlog changes and the team's velocity—estimate cost top-down only based on your product roadmap. This will not only save you time and money, but it also avoids creating an overly long and complex product backlog that is difficult to adjust and maintain.

Admittedly, this approach will result in a rough, high-level estimate. But my experience suggests that a bottom-up approach does not produce a more robust figure. It just appears to be more precise. But as the economist John Maynard Keynes said, "it is better to be roughly right than to be precisely wrong."

Flexible Budget

If you can acquire the necessary budget, then determine how many people are likely to be required to implement the product roadmap and which skills they will have to have. To do this, draw on the knowledge the development team members have acquired while helping validate the product strategy and on their experience of developing similar products or previous versions of the same product. This will allow you to come up with a rough estimate of the labour cost.

Add the cost for facilities, equipment, licences, and other relevant items and you'll get a high-level cost estimate. To avoid the impression that it is precise, state it as a range, for instance, £350–400K. If the resulting figure is too high, or if not enough people with the right skills are available and cannot be hired, then you have two options: persevere and adjust your roadmap or pivot and search for a more cost-effective and feasible strategy.

Don't forget to track the development progress and the actual expenditures once development has started. This will help you understand if the initial cost estimate was realistic and if that's not the case, make the necessary changes.

Fixed Budget/Development Team

If your budget has been pre-allocated and you cannot increase the size of an existing development team and add new teams, then ask the team members whether they are confident that they can meet the product goals on the roadmap within the desired time frames, without putting any pressure on them. If the answer is no, then adjust your roadmap; make the product goals less ambitious or extend the time frames. Iterate over the plan until it has become feasible, and it can be put into action by following a sustainable pace.

Balance Product Goals, Dates, and Cost

In theory, you'd liked to meet all product goals on the roadmap on time and on budget. But in practice, that's not always possible. For example, technical challenges might prevent you from creating the desired outcome at the right date, or you might not be able to hire enough people with the right skills. Balancing goals, dates, and cost helps you maximise the value your product creates and at the same time, ensure that your roadmap is realistic and actionable—that it can be implemented without sacrificing sustainable pace or incurring technical debt.[81]

Iron Triangle

A handy tool for illustrating that product goal, time, and cost cannot be fixed simultaneously is the *Iron Triangle*. The triangle shown in Figure 28 connects three key factors, product goal, time, and cost.[82] Each factor forms one of the triangle's vertices.

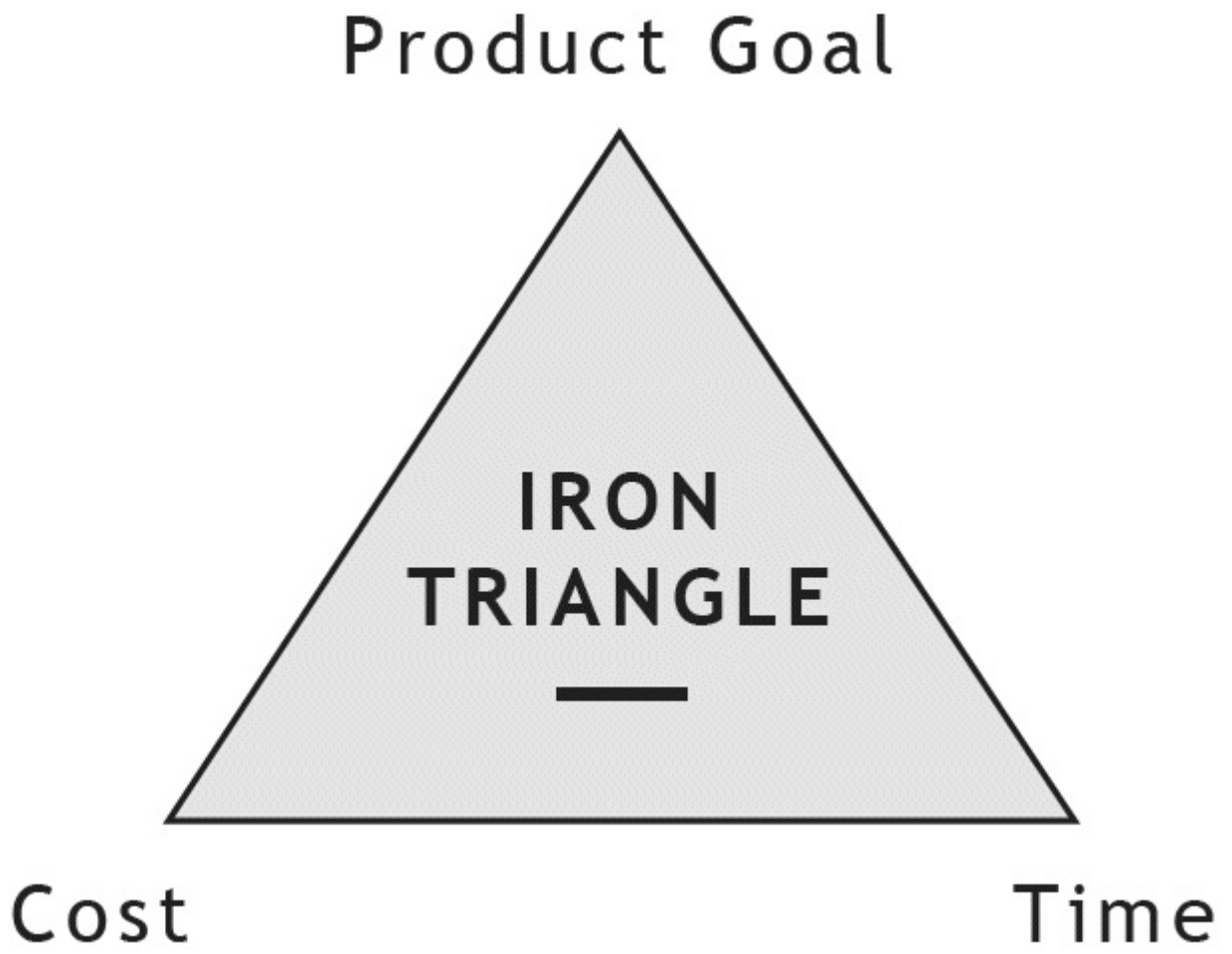


Figure 28: The Iron Triangle Revisited

The Iron Triangle states that at least one of its corners must stay flexible and act as a release valve to account for unforeseen events. You cannot lock down all three factors. Otherwise, you risk that quality is compromised and people start working overtime—neither of which is desirable. As mentioned before, compromising quality will make it harder to change and enhance your product in the future. Working overtime is likely to lead to lower productivity, reduced motivation, and, in the worst case, health issues if it carries on for too long. I know several people who suffered from heart attacks, slipped discs, and lasting food intolerances because they experienced high stress levels in software development for an extended period. No product is worth paying this price. Therefore, fix the software quality and accept that the roadmap will be implemented using regular working hours.[83]

But if not all vertices can be locked, then which ones should you fix? Which of the three factors is the most important one? To answer this question, you will need to determine the primary success factor.

Primary Success Factor

The primary success factor is the element that has the biggest impact on the success of the product—fully meeting the goal, releasing software on time, or adhering to the budget (Kerzner 2013). Sometimes, the primary success factor is obvious. If you know, for example, that your product must be ready for the Christmas sales or a major trade show, then on-time delivery is a must and therefore the primary success factor. But if you are unsure, then carry out an impact analysis. Ask yourself if partially meeting the product goal, releasing late, or overrunning the budget would have the worst impact on the product's ability to create value.

Say that I am going on a road trip with my family, and we have pre-booked a hotel to stay in overnight. Spending the night in the hotel would then be my primary success factor. Here is why: Not making it to the hotel and having to sleep in the car would be worse than arriving late or increasing our travel budget, for instance, by using faster toll roads.

Once you have found the primary success factor, you should protect it. If it's on-time delivery, for example, then do everything you can to ship your product on schedule. Consider the first iPhone. To release the product on time, Apple was prepared to partially implement some features and to increase cost by adding more people to the development effort. The very first release shipped without the ability to send text messages to multiple recipients, for example, a feature every ordinary mobile phone offered. As this example shows, protecting the primary success factor requires you to relax at least one other factor.

Secondary Success Factor

Having determined the primary success factor is great. But often, it's not enough, and you will benefit from identifying the secondary success factor. As its name suggests, this factor has the second biggest impact on the value the product creates.

Take the road trip example used earlier. If getting to the hotel is my primary success factor, then it is still helpful to know if I should arrive on time or rather adhere to my travel budget, as this will allow me to make the right driving choices. To find out which one is more important, I would consider the impact of arriving late and missing dinner and compare it to spending more money to speed up the journey and get to the hotel faster. Personally, I'd rather exceed the travel budget than sleep with a rumbling tummy. This makes getting to the hotel on time the secondary success factor.

As it is not as crucial as the primary one, you should protect the secondary success factor as much as possible. Be willing, though, to flex it to a certain extent. Applying it to the road trip example, this may mean that I'll arrive later than desired but still in time for dinner to be served.

Fixed vs Changing Success Factors

Sometimes, the primary and secondary success factors apply to the entire roadmap. Other times, they don't. For a new product development effort, for example, you might have to fully meet the initial product goal (i.e., the primary factor) and do your best to release the

product within a certain time frame (the secondary factor). For the remainder of the roadmap, however, delivering on time might become the primary success factor, and fully reaching the product goals the secondary one.

Note that balancing the product goals, the dates, and the budget may require you to iterate your product roadmap, particularly when a significant amount of uncertainty or change is present until you have found an acceptable trade-off.

Derive the Product Backlog from the Roadmap

Before I start a road trip, I usually enter my destination into a mapping tool and select the route I want to use. Based on this choice, the app offers me detailed instructions that guide me while I am driving. Similarly, your product backlog should be based on the roadmap: It should contain the work necessary to meet the product goals you have set.

You can take this approach further and focus your product backlog on the next product goal. To do this, carry out the following four steps together with the development team members:[84]

1. Copy the next product goal from your product roadmap into your product backlog.
2. Remove any items that are not required to meet the goal.
3. Copy the features that belong to the product goal from the roadmap to the backlog. Then ask yourself which additional items are required to meet the goal and add them to the product backlog.
4. Prioritise the backlog, break down the high-priority items and refine them so that they are ready to be implemented.

This approach offers two benefits: First, it systematically connects the product roadmap and the product backlog: The latter is derived from the former. Second, it significantly reduces the product backlog size by focusing its contents on one product goal at a time. Such a backlog is easier to stock, prioritise, refine, and update compared to one that covers, say, the next twelve months. This is particularly helpful when you are faced with uncertainty, risk, and innovation, for example, when your product is brand-new or young, or when you are about to make a bigger change to it like adding new features, adjusting the business model, or changing the technology stack.[85]

Be aware, though, that the connection between the product roadmap and the product backlog is bidirectional: The backlog can also influence the roadmap. Bigger backlog changes may trigger roadmap adjustments. For instance, if the remaining effort in the product backlog grows based on the user feedback you receive, then this may indicate that the goal cannot be met as expected. Consequently, you may have to adjust it. This might have a knock-on effect causing other product goals or dates on the roadmap to be updated. Similarly, if the development work does not progress as quickly as anticipated, you may have to adapt the product roadmap and modify, for example, the goal or the date. It is therefore important that you keep the product roadmap and the product backlog in sync, as I explain in more detail in the chapter *Product Roadmap Reviews*.

Dealing with an existing product backlog

If you find yourself in a situation where you have inherited an existing product backlog but lack a product roadmap, then you have two choices: reverse-engineer the roadmap from the backlog or start from scratch and discard the existing product backlog.

In the first case, you might group related backlog items, identify the outcomes you might be able to achieve by implementing them, and then use the outcomes to derive a goal-oriented roadmap. In the second case, you would derive a product roadmap from the product strategy—assuming that a validated strategy exists, and then follow the four steps above to create a new product backlog. Once you've done this, you can use the old product backlog to check if you have missed anything significant and if that is the case, adjust the new backlog. Afterwards, delete or archive the old one.

My preference is to opt for the second approach and derive a new product backlog from the roadmap. This ensures that the two plans are effectively connected, that the new backlog is concise, and that its items are required to achieve the product goal selected.

Align Related Products with a Portfolio Roadmap

Some products have—due to their very nature—close connections to others. This is particularly true for supporting products and internal software platforms. If that's the case for your product, then you may benefit from using a portfolio roadmap: a plan that shows how a group of related products will evolve together. Using a portfolio roadmap can make it easier to coordinate the development and release of related products, and it can help you identify and manage dependencies between them. Figure 29 shows my preferred version of a portfolio roadmap, which is based on the GO roadmap template.

The GO portfolio roadmap in Figure 29 combines the roadmaps of several related products into a single goal-oriented plan. Its top section displays the shared portfolio dates or time frames. It then states the products that form the portfolio: product A and product B. Each product has its own goals, features, and metrics. You can add more than two products to a portfolio roadmap, of course. Be careful, though, to keep the plan readable and easy to understand. Portfolio roadmaps with more than five products can be overwhelming and hard to comprehend due to the amount of information they offer.

DATE/TIME FRAME					
PORTFOLIO α	PRODUCT A				
	Goal				
	Features				
	Metrics				
	PRODUCT B				
	Goal				
	Features				
	Metrics				

Figure 29: The GO Portfolio Roadmap

If you have a larger portfolio, then you may want to use a portfolio roadmap that contains only those products that have especially close connections and complement it with individual product roadmaps for the other portfolio members. Say that you manage a product portfolio like Microsoft Office and that you want to align its core members. I would then recommend creating a portfolio roadmap for Word, PowerPoint, and Excel, to stay with the example, and individual roadmaps for the other products, such as Outlook and Teams.

Finally, if your product is constrained by a large number of dependencies, then recognise that a portfolio roadmap may not be sufficient. You may also have to break some of the dependencies along the lines discussed in the section *Prioritise the Product Goals*.

Leverage Collaborative Workshops

A great way to involve the stakeholders and development team members in the roadmapping work is to use collaborative workshops—be they online or onsite—along the lines described in the chapter *Product Strategy Foundations*.

Running such a workshop offers the following three benefits: First, it helps you make better decisions by leveraging the collective creativity and expertise of the attendees. This helps you create a roadmap that maximises the value the product creates and that is

realistic and actionable.[86] Second, it creates a shared understanding and aligns the stakeholders and development team members with the product goals. Third, it results in stronger buy-in: Inviting people to contribute to a decision tends to generate more support, and it increases the chances that people will implement the roadmap and follow its goals.

Start the workshop by briefly reviewing the product strategy. Ideally, the participants were involved in creating and testing the strategy, and they are already familiar with it. Then follow the practices described previously in this chapter: Select the right product goals, make the goals specific and measurable and order them; derive key features; determine dates and cost; and finally balance goals, dates, and cost. Additionally, make sure that you use the right collaborative decision-making techniques, including the following four methods, which are based on Pichler (2020).[87]

- *Ask your Scrum Master to facilitate the workshop*. This is especially beneficial when the players don't know and trust each other yet and when people aren't familiar with collaborative decision-making. Additionally, it allows you to focus on contributing to the roadmap rather than ensuring that everyone is heard, and nobody dominates.
- *Use consent as the decision rule*. This means that a decision is made when nobody disapproves, and no meaningful objections are raised. I find that consent is comparatively quick to achieve and that it usually generates sufficient agreement for roadmap decisions.
- *Lead by example*. Listen to the attendees attentively while keeping an open mind. Use empirical evidence to make decisions instead of gut feeling or the highest paid persons' opinion (HIPPO).
- *Don't rush the roadmapping work*. Spend enough time especially on setting the right product goals—they are the backbone of an effective goal-oriented, outcome-based product roadmap.

Using the techniques above will help you make the right decisions and at the same time, secure as much buy-in from the stakeholders and dev team members as possible. Additionally, it will mitigate the risk that individuals dominate and dictate the roadmap contents or that you end up brokering a weak compromise and agree on the smallest common denominator—neither of which is likely to result in an effective product roadmap.

Allocate two to four hours for the workshop. If you need significantly more time, then this may indicate that you haven't carried out the necessary discovery work, that your product strategy hasn't been sufficiently validated, or that you don't apply the right decision-making techniques.

PRODUCT ROADMAP REVIEWS

Intelligence is the ability to adapt to change.
Stephen Hawking

It can be tempting to view the product roadmap as a hard-and-fast plan that simply needs to be executed well. But this would be wrong. Product-development efforts don't always go according to plan, and unforeseen things do happen. As Murphy's Law states: "Anything that can go wrong, will go wrong."[88] The progress might not be as fast as anticipated, for instance, the effort to reach a product goal might be higher than originally estimated, or one of the technologies might cause issues. As these challenges can impact the product roadmap, you should review and update the plan on a regular basis.[89]

Track the Development Progress

Say that I've carefully mapped out the route to my holiday destination. But despite my best efforts, I'll only find out if my plan holds true once I have started the journey. I might discover that the traffic is free flowing, and I progress faster than anticipated; or I might get stuck in slow traffic and experience a delay. The same is true for your product roadmap. Only after you have started working on a product goal and you are able to measure the development progress will you find out if the plan is correct.

A handy tool to track and forecast the development progress is the *release burndown chart* shown in Figure 30.[90]

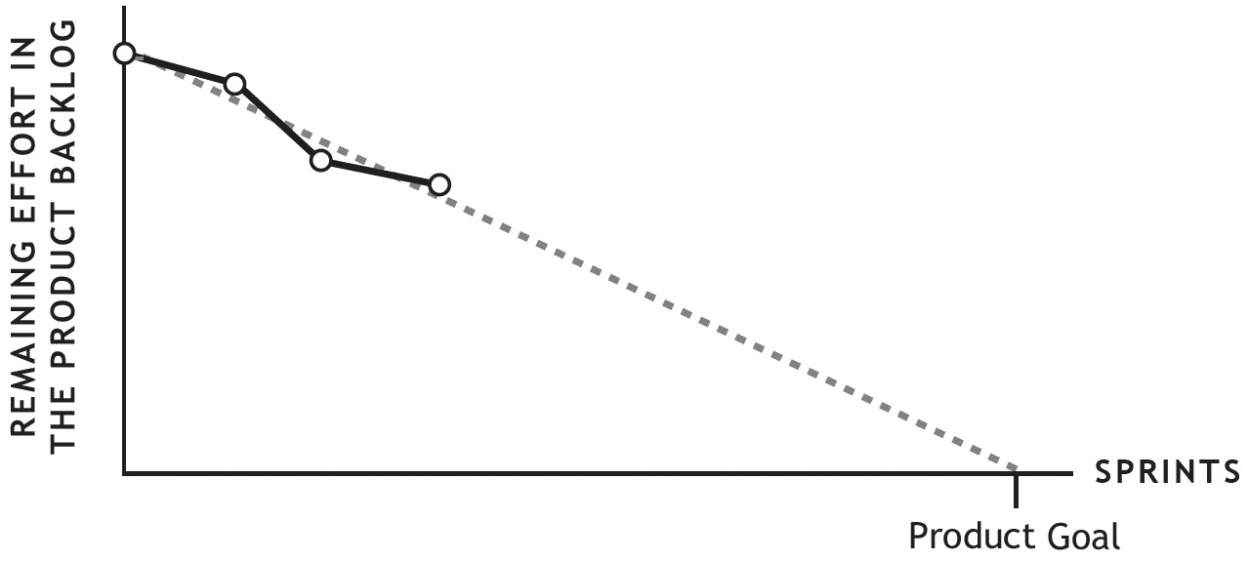


Figure 30: Release Burndown Chart

The horizontal axis of the release burndown chart in Figure 30 shows the sprints; the vertical axis states the remaining work in the product backlog. The first data point on the chart is the estimated effort in the backlog before any development has taken place.[91] To arrive at the next data point, determine the remaining effort in the product backlog at the end of the first sprint, after you have assessed which product backlog items are done. Then draw a line between the two points. This line is called the *burndown*. Note that the burndown of the third sprint is slower than the burndown of the second one. This might be caused by a drop in velocity, as team members went on holiday, fell ill, or attended a training course, for example. But it could also be due to product backlog changes: Based on the feedback received on the latest product increment, new items might have been added to the backlog. Alternatively, the development team members might have corrected some of the effort estimates. Consequently, the effort in the product backlog increased.

Once the burndown covers a few sprints, you should see a trend emerge and be able to forecast future progress. This forecast is represented by the dotted line in Figure 30. Luckily, this line indicates that everything is on track, that the backlog items can be delivered within the desired time frame, and that the product goal can be met as planned. But that's not always the case. If the forecast predicts that the goal cannot be fully met on time, then recognise that the development effort is currently off track and investigate the underlying causes together with the development team members. It might turn out that the team lacks manpower or skills, that some of the technologies cannot be used as expected, or that more features have to be implemented than you had anticipated. Once you have determined the causes, use your primary success factor, and take the right action. For instance, you might decide to weaken the goal, push out the date, or increase the budget. Then update the product roadmap, along the lines described in the next section.

Inspect and Adapt the Roadmap

To effectively review and update your product roadmap, you need to take into account the right data and carry out the reviews at the right frequency. Let's look at these two aspects starting with the review factors, the data that helps you assess if the roadmap is valid.

Review Factors

There are three factors that help you determine if the product roadmap needs to be adjusted: product strategy changes, work progress, and product backlog changes.

1. *Product strategy changes*: When the product strategy is modified, the roadmap usually needs to change too. Especially the product goals will have to be replaced or reworked. This ensures that the product roadmap continues to be an actionable plan that is aligned with the strategy.[92]
2. *Progress*: Consider how well the work on the product has progressed. Is the

development effort on track, and are you likely to meet the current product goal on time and on budget? And how is the work of the stakeholders coming along? For example, will the marketing and sales collateral be available on time? If the progress is slower or faster than expected, then check if the roadmap is still valid. If that's not the case, adapt it. You may have to adjust the goals and dates, as well as the metrics and features.

3. *Product backlog changes*: Check if there are any bigger product backlog alterations. Did you add, remove, or significantly change existing product backlog items based on the user feedback you have received? Do these changes require a roadmap update? For instance, do you have to adjust any of the goals or their key features? Carrying out this work helps you keep the roadmap and backlog in sync. It avoids that the two plans diverge and become inconsistent.

Note that insights from the execution—from developing the actual product, measuring the progress, collecting user feedback, and adapting the backlog—can trigger product roadmap changes. If these are significant, they may cause a change of the product strategy. Strategy and execution are therefore connected, as pointed out in the *Introduction*. Strategy guides execution, and execution informs strategic product decisions.

Brook's Law

To meet a goal on time, you might be tempted to add more people to the development effort and increase the budget. But scaling should be planned carefully, and not be a fire-fighting measure. As Brooks' law states, "Adding manpower to a late software project makes it later." (Brook 1995) There are two reasons for this: First, the new members usually need time and help to get up to speed, and second, changing a team tends to negatively impact the group dynamics. Both factors initially cause a drop in productivity. When the development progress is slow, your best option might therefore be to trade off product goal and date—instead of increasing the budget and adding more people.

Review Frequency

To ensure that your product roadmap stays valid and actionable, I recommend the following dual approach: Use sprint-end reviews and quarterly strategic reviews to assess the plan.

1. *Sprint-end reviews*: Consider the latest development progress and user feedback at the end of each sprint. Assess if the current product goal can be met on time and on budget. If that's not the case, then determine the impact on the product roadmap and adapt the plan along the lines described above. To avoid overhead, carry out this work as part of the regular sprint review meetings.
2. *Quarterly reviews*: Assess the product roadmap together with the product strategy once every three months. Combining the strategy and roadmap reviews saves you scheduling an additional meeting, and it ensures that the two plans stay closely aligned. (See the section *Regularly Review and Update the Product Strategy* in the

chapter *Product Strategy* Reviews for more information).

Note that sprint-end reviews are especially helpful when you face a significant amount of uncertainty and change, for example, when your product is young or when you make a bigger change to an existing offering. Once your product is experiencing a phase of stability or it has entered the growth stage, you might decide to skip sprint-end reviews and only employ quarterly ones, at least for the time being. But don't make the mistake of waiting for the next quarterly review if a bigger roadmap change is urgently needed. Instead, schedule a review as soon as possible so you can collaboratively adapt the plan.

The Sprint Review Meeting in a Nutshell

The sprint review meeting is possibly the most important Scrum event for product people: It helps you understand what has been achieved in a sprint and collect feedback on the latest product increment. The former allows you to create, or update, the release burndown chart, understand if the development progress is on track, and realign the development teams and key stakeholders—who should regularly attend the meeting.

Collecting user feedback helps you validate the product increment and the detailed product decisions you have made, discover new opportunities to change and improve the product, and update the product backlog and the product roadmap, if required. This assumes, though, that (selected) users and customers are present at the meeting and a method like product demo is used to collect the feedback. For more information on how to effectively use the meeting, see Pichler (2017).

Involve the Right People and Hold the Individuals Accountable

It probably doesn't come as a surprise that I recommend involving the key stakeholders and development team members in reviewing and updating the product roadmap. This allows you to leverage their expertise and make the right roadmapping decisions, to create a shared understanding, and to maximise their buy-in. A great way to engage the individuals is to invite them to the quarterly reviews and hold them as collaborative workshops along the lines discussed in the previous chapter. Be careful, though, not to say yes to every idea or request. This would turn the roadmap into a wish list and possibly result in a Frankenstein product with botched-up features and a weak value proposition. Do listen to the individual's request with an open mind, empathise with them, and try to understand their underlying needs and interests. But have the courage to say no if the request is not in line with the product strategy or the cost of changing the roadmap outweighs the benefit the feature might offer.

Additionally, don't forget to hold the stakeholders and development teams accountable for meeting the goals on the product roadmap, assuming that they have agreed to them. Imagine that the people who travel with you turn up late for a road trip, forget to bring their luggage, or continuously complain about the route choice. I guess you would then offer candid feedback and ask them to change their behaviour. Likewise, if, for example, the marketer fails to carry out the necessary work, then don't be afraid to address the issue. It is not acceptable that a stakeholder or team member intentionally goes against a joint decision or does not follow an agreed goal.

If someone repeatedly and knowingly goes against the decisions captured on the product roadmap despite having been involved in the decision-making process, and if you have talked to the individual and tried to amicably resolve the issue, consider asking the person to leave the group. In any case, do not put up with toxic behaviour, and don't shy away from difficult conversations. Untreated people problems seldom go away on their own. Instead, they usually grow bigger, hamper productivity, and make it difficult to interact with the individuals.

Addressing Development Team Issues

If you are unhappy with the work of the development team, then address the issue in the next sprint retrospective. Offer candid feedback and share your perspective—but without judgement and blame. Cultivate an open mind and actively listen to the suggestions of the development team members. You might discover, for example, that slow development progress was caused by big and unclear product backlog items as well as a lack of team member involvement in the backlog refinement work. These causes can only be fully resolved if you are willing to change the way you work.

If an issue persists despite determining its causes and implementing improvement measures, then consider further steps. For instance, it might be beneficial to break up a team whose members work on separate areas of the same product and have started to form two sub teams. In some cases, it might also be necessary to ask someone who continuously shows disruptive behaviour to leave the team. However, none of these changes should be forced onto the team. Instead, be inclusive and look for a solution that works for everyone. This creates transparency, gives people the opportunity to contribute, and it reduces the risk that individuals end up frustrated and feeling treated unfairly, as I discuss in more detail in Pichler (2020).

EPILOGUE

Your present circumstances don't determine where you can go. They merely determine where you start.
Nido Qubein

I hope that you found the concepts, techniques, and tools covered in this book helpful. But just like strategy without execution is of limited benefit, so is reading this book and not applying the insights you have gained. I would therefore like to encourage you to put your new knowledge to work. This does not necessarily mean that you will have to fundamentally change the way your organisation makes strategic product decisions. Sometimes, a small first step can lead the way to profound changes. Such a step might be reaching out to (prospective) users and talking to them rather than exclusively relying on analytics data and input from the sales team; it might be starting to work with a goal-oriented product roadmap; or it might be adopting a collaborative approach and involving the key stakeholders and development team members in strategic product decisions. As Lao Tzu said, a journey of a thousand miles begins with a single step.

ACKNOWLEDGMENTS

This book would not have been possible without the help and support of many people. I am particularly grateful to my wife, Melissa Pichler, who helped me with the first and the second edition of this book including discussing ideas and reviewing the manuscripts. I would also like to thank Petra Färm for helping me decide how to structure the second edition and everyone who offered feedback on the first edition, especially Marc Abraham and Stefan Roock.

ABOUT THE AUTHOR

Roman Pichler works as a product management consultant, trainer, and writer. He has a long track record of teaching and mentoring product managers and owners and advising product leaders. He is the author of three other books, including *Agile Product Management with Scrum* and *How to Lead in Product Management*. He writes a popular product management blog and hosts his own podcast. Roman lives with his wife and three children near London in the United Kingdom. You can find out more about his work and contact him at www.romanpichler.com.

REFERENCES

Anderson, David. 2010. *Kanban*. Blue Hole Press.

Andreessen, Marc. 2007. "The only thing that matters." The PMARCA guide to startups, 25 June. Accessed on 27 March 2021. https://pmarchive.com/guide_to_startups_part4.html

Ansoff, Igor 1957. "Strategies for Diversification." *Harvard Business Review* 35 (5): 113–24.

Arnold, Joshua. 2016. "Qualitative Cost of Delay." Black Swarm Farming. Accessed 20 September 2021. http://blackswanfarming.com/qualitative-cost-delay/

Baker, Michael and Susan Hart. 2007. *Product Strategy and Management*. 2nd Edition. Financial Times/Prentice Hall.

Beck, Kent. 2000. Extreme Programming Explained. Embrace Change. Addison-Wesley.

Bland, David J. and Alexander Osterwalder. 2019. *Testing Business Ideas: A Field Guide for Rapid Experimentation*. Wiley.

Blank, Steve. 2014. "The Key to Startup Success: Get out of the Building." *Inc. Video The Playbook*, 2 November. Accessed 18 February 2022. http://www.inc.com/steve-blank/key-to-success-getting-out-of-building.html

Brignull, Harry. 2021. "Type of Dark Patterns." Accessed 22 February 2021. https://www.darkpatterns.org/types-of-dark-pattern

Brooks, Frederick P. 1995. *The Mythical Man Month and Other Essays on Software Engineering*. Second Edition. Addison-Wesley.

Brown, Tim. 2009. *Change by Design*. HarperBusiness.

Cagan, Marty. 2018. *Inspired: How to Create Tech Products Customers Love*. 2nd edition. Wiley. Kindle Edition.

Case, A. *Calm Technology: Designing for Billions of Devices and the Internet of Things*. Boston: O'Reilly Media, 2015.

Christensen, Clayton. M. 1997. *The Innovator's Dilemma: When New Technologies Cause Great Firms to Fail*. Harvard Business School Press.

Christensen, Clayton M. and Michael E. Raynor. 2013. *The Innovator's Solution, Revised and Expanded: Creating and Sustaining Successful Growth*. 2nd Ed. Harvard Business Review Press.

Cohn, Mike. 2005. *Agile Estimating and Planning*. Prentice Hall.

Collins, James. 2005. *Built To Last: Successful Habits of Visionary Companies*. Random House Business.

Cooper, Alan. 1999. *The Inmates Are Running the Asylum*. Sams Publishing.

Coyne, Kevin. 2008. "Enduring Ideas: The GE–McKinsey nine-box matrix." McKinsey Quarterly, September. Accessed 20 September 2021. https://www.mckinsey.com

/business-functions/strategy-and-corporate-finance/our-insights/enduring-ideas-the-ge-and-mckinsey-nine-box-matrix

Croll, Alistair and Benjamin Yoskovitz. 2013. *Lean Analytics. Use Data to Build a Better Startup Faster*. O'Reilly Media.

Cunningham, Ward. 1992. "The WyCash Portfolio Management System." *OOPSLA '92 Experience Report*. Accessed 18 November 2021. http://c2.com/doc/oopsla92.html

Doerr, John. 2018. *Measure What Matters: OKRs: The Simple Idea that Drives 10x Growth*. Portfolio Penguin.

Downes, Larry and Paul Nunes. 2013. "Big-Bang Disruption." *Harvard Business Review* (May): 44–56.

Drucker, Peter F. 1985. *Innovation and Entrepreneurship*. Harper & Row.

Eden, Colin and Fran Ackermann. 2011. *Making Strategy: Mapping Out Strategic Success*. 2nd Ed. SAGE Publications.

Gray, Dave. 2017. "Updated Empathy Map Canvas." Medium. Accessed on 23 November 2021. https://medium.com/the-xplane-collection/updated-empathy-map-canvas-46df22df3c8a

Henderson, Bruce. 1970. "The Product Portfolio." BCG Publications. Accessed on 15 March 2021. https://www.bcg.com/publications/1970/strategy-the-product-portfolio

Islam, Nazrul and Sercan Ozcan. 2012. "Disruptive Product Innovation Strategy: The Case of Portable Digital Music Players." In *Disruptive Technologies, Innovation and Global Redesign: Emerging Implications*. Edited by Ndubuisi Ekekwe and Nazrul Islam, 27–45. IGI Global.

Kano, Norikani. 1984. "Attractive Quality and Must-Be Quality." *Journal of the Japanese Society for Quality Control* (April): 39–48.

Kenny, Graham. 2016. "Strategic Plans Are Less Important than Strategic Planning." Harvard Business Review, 21 June. Accessed 7 July 2021. https://hbr.org/2016/06/strategic-plans-are-less-important-than-strategic-planning.

Kerzner, Harold R. 2013. *Project Management: A Systems Approach to Planning, Scheduling, and Controlling*. 11th Ed. John Wiley & Sons.

Kim, W. Chan and Renée A. Mauborgne. 2004. *Blue Ocean Strategy*. Harvard Business Review Press.

Knapp, Jake, John Zeratsky, and Braden Kowitz. 2016. *Sprint: How To Solve Big Problems and Test New Ideas in Just Five Days*. Bantam Press.

Lapowsky, Issie. 2013. "Ev Williams on Twitter's Early Years." *Inc.com*, 4 October. Accessed 28 September 2021. www.inc.com/issie-lapowsky/ev-williams-twitter-early-years.html.

Levitt, Theodore. 1965. "Exploit the Product Life Cycle." *Harvard Business Review* 43: 81–94.

Liker, Jeffrey. 2004. *The Toyota Way: 14 Management Principles from the World's Greatest Manufacturer*. McGraw-Hill Professional.

Martin, Roger. 2013. *Playing to Win: How Strategy Really Works*. Harvard Business Review Press.

Maurya, Ash. 2012. *Running Lean. Iterate from Plan A to a Plan That Works*. 2nd Ed. O'Reilly Media.

MacMillan, Ian and Rita Gunther McGrath. 1997. "Discovering New Points of Differentiation" *Harvard Business Review* (July): 133-8.

Moon, Youngme. 2005. "Break Free from the Product Life Cycle." *Harvard Business Review* (May): 86–94.

Moore, Geoff. 2006. *Crossing the Chasm. Marketing and Selling Disruptive Products to Mainstream Customers*. Collins Business Essentials.

Nagji, Bansi, and Geoff Tuff. 2012. "Managing Your Innovation Portfolio." *Harvard Business Review* (May): 66–74.

Norton, David P. and Robert S. Kaplan. 1996. *The Balanced Scorecard: Translating Strategy into Action*. Harvard Business School Press.

Osterwalder, Alexander and Yves. Pigneur. 2010. *Business Model Generation*. John Wiley & Sons.

Osterwalder, Alexander and Yves. Pigneur. 2014. *Value Proposition Design: How to Create Products and Services Customers Want*. Wiley.

Pichler, Roman. 2010. *Agile Product Management with Scrum: Creating Products that Customers Love*. Addison-Wesley.

Pichler, Roman. 2016. "Scaling the Product Owner Role." 28 June. Accessed 8 July 2021. https://www.romanpichler.com/blog/scaling-the-product-owner

Pichler, Roman. 2017. "Sprint Review Tips for Product Owners." 6 November. Accessed 8 July 2021. https://www.romanpichler.com/blog/sprint-review-tips-for-product-owners/

Pichler, Roman. 2020. *How to Lead in Product Management: Practices to Align Stakeholders, Guide Development Teams, and Create Value Together*. Pichler Consulting.

Porter, Michael E. 1996. "What Is Strategy?" *Harvard Business Review* (November-December 1996): 61-78.

Ries, Al and Jack Trout. 1994. *The 22 Immutable Laws of Marketing*. Profile Books.

Ries, Eric. 2009a. "Vanity Metrics vs. Actionable Metrics." *The Tim Ferriss Experiment*, 19 May. Accessed 18 February 2022. http://fourhourworkweek.com/2009/05/19/vanity-metrics-vs-actionable-metrics/.

Ries, Eric. 2009b. "Pivot, don't jump to a new vision." *Startup Lessons Learned*, 22 June. Accessed 1 Mar 2022. http://www.startuplessonslearned.com/2009/06/pivot-dont-jump-to-new-vision.html.

Ries, Eric. 2009c. "Minimum Viable Product: a product guide." *Startup Lessons Learned*, 3 August. Accessed 16 March 2021. http://www.startuplessonslearned.com/2009/08/minimum-viable-product-guide.html.

Ries, Eric. 2011. *The Lean Startup: How Constant Innovation Creates Radically Successful Businesses*. Portfolio Penguin.

Rowe, Ben. "The Rise of the Ethical Product Designer." December 23, 2018. Accessed 3 November 2021. http://www.uxmas.com/2018/the-rise-of-design-ethics.

Torres, Teresa. 2021. *Continuous Discovery Habits: Discover Products that Create Customer Value and Business Value*. Product Talk LLC. Kindle Edition.

Ulwick, Anthony W. 2016. *Jobs to be Done: Theory to Practice*. Idea Bite Press. Kindle Edition.

Womack James P. and Daniel T. Jones. 2005. *Lean Solutions. How Companies and Customers Can Create Value Together*. Simon & Schuster.

Wysocki, Robert K. 2013. *Effective Project Management: Traditional, Agile, Extreme*. 7th Ed. Wiley.

NOTES

1 I use the term *development team* in this book to refer to a cross-functional, self-managing group whose members design, architect, implement, test, document, and deploy a digital product.

2 In the first edition of this book, I used the term *release goals* instead of product goals. But as not every release necessarily creates tangible value for the users and the business, for instance, a smaller bug-fixing release, I have changed the name. This brings the term in line with my more recent work (Pichler 2020) and the Scrum Guide 2020. The latter recognises product goals as part of the Scrum framework.

3 A *product increment* is the output of a sprint. It is commonly defined as working software that has been tested and documented and that might be released. You can think of an increment as a reusable prototype, a stepping stone towards a new product or a new product version.

4 Figure 2 offers a process-oriented view of the model shown in figure 1. It complements the strategy and roadmap model with additional elements like product increment, product, development progress, and KPIs. Please note, though, that Figure 2 simplifies some of the connections to clearly illustrate the relationship between strategy and execution. For example, the figure that does not show that bigger product backlog changes can trigger a roadmap modification or that development progress data can lead to a roadmap update.

5 The model in Figure 1 implements the hierarchy of product-related goals I propose in Pichler (2020). I explain in more detail how you can discover the right needs and business goals in the chapter *Product Strategy* Development, and I'll discuss how you can derive product goals from the needs and business goals in the chapter *Product Roadmap* Development.

6 Product discovery refers to the activities required to determine if and why a product should be developed. Carrying out this work makes it more likely to create a product that users want and need. Cagan (2018, 27) suggests that this involves answering the following four questions: Will people use or buy the product? Can the users figure out how to use it? Can the developers build it? Can the stakeholders support the product? As this definition shows, product discovery and strategy work can be viewed as largely identical.

7 Torres (2021, 21) suggests that continuous discovery involves carrying out research activities at least once per week, which include interacting with customers.

8 One way to achieve this is to ask the person in charge of the product to look after the product strategy and product roadmap as well as the product backlog. This results in what I call full-stack ownership: The individual owns the product top to bottom, so to speak. To apply this concept on larger products and in a scaled environment, use specialised product roles that are responsible for product parts like features and components as I explain in Pichler (2016). Using product goals on the product roadmap and copying the next one into the product backlog is another alignment measure, as I discuss in more detail in the section *Derive the Product Backlog from the Roadmap*.

9 Taking on Scrum Master responsibilities does not only increase your workload. It also hides a systemic problem and organisational impediment: If you seem to be able to take on the Scrum Master role, then there is little need for your company to hire or develop Scrum Masters.

10 For more information on product bundles, please refer to the section *Create a Product Bundle* in the chapter *Product Strategy* Development.

11 Please see the section *Take Advantage of Software Platforms* in the chapter *Product Strategy* Development for more information on platforms.

12 As these examples show, I view features as product capabilities. Features are broken into epics in the product backlog. You can also think of a feature as a group of epics, which is also referred to as a *theme*.

13 See https://www.tesla.com/en_GB/about.

14 The three options were originally suggested by Michael Porter (Porter 1996).

15 See Nagji and Tuff (2012). Note that I use the term *disruptive* instead of *transformational*, which Nagji and Tuff (2012) employed. Some people use *incremental* instead of *core*, *evolutionary* for *adjacent*, and *revolutionary* or *breakthrough* for *disruptive*.

16 The Innovation Ambition Matrix is based on the Ansoff matrix, which explores the relationship between the product and the market; it distinguishes an existing product from a new product and an existing market from a new one. This gives rise to four growth strategies: market penetration, product development, market development, and diversification. *Market penetration* means incrementally enhancing an existing product to increase its market share. *Product development* involves creating a new product for an existing market—a market you already serve. *Market*

development refers to entering a market that's new to your company with an existing product. *Diversification* implies developing a new product for a new market (Ansoff, 1957).

17 Facebook added a "send money" feature to its Messenger product in March 2015. For more information, see http://newsroom.fb.com/news/2015/03/send-money-to-friends-in-messenger/

18 Christensen (1997) refers to core and adjacent innovations as *sustaining*, as they address established markets and build on existing assets.

19 Christensen and Raynor (2013) argue in Part 4 of their book that an incubator is mandatory for succeeding with disruptive innovations.

20 Christensen (1997, p. 143)

21 Nagji and Tuff (2012) recommend that companies should invest at least 10 percent in disruptive innovations.

22 Theodore Levitt (1965) first described the product life cycle model in his article "Exploit the Product Life Cycle." For comprehensive discussion of the product life cycle, refer to Baker and Hart (2007).

23 The term *product-market fit* was introduced by Andreessen (2007) who suggests the following heuristics to define it: "The customers are buying the product just as fast as you can make it—or usage is growing just as fast as you can add more servers. Money from customers is piling up in your company checking account. You're hiring sales and customer support staff as fast as you can. Reporters are calling because they've heard about your hot new thing and they want to talk to you about it." This description seems to support my view, which equates product-market fit with entering the growth stage.

24 The sources for the data shown in the graph include

- https://commons.wikimedia.org/wiki/File%3AIpod_sales_per_quarter.svg
- statista.com/statistics/276307/global-apple-ipod-sales-since-fiscal-year-2006/
- apple.com/pr/products/ipodhistory/.

25 See, for example, Ash Maurya's article "What is a Minimum Viable Product (MVP)?", http://ask.leanstack.com/en/articles/902991-what-is-a-minimum-viable-product-mvp. I refer to the initial release of a product as the *minimal marketable product* in my book *Agile Product Management with Scrum* (Pichler 2010, pp. 27).

26 The quote is from Lapowsky (2013).

27 Google has released most of Wave's source code to the Apache Software Foundation: https://en.wikipedia.org/wiki/Apache_Wave and http://incubator.apache.org/wave/about.html.

28 In his book *Crossing the Chasm*, Geoffrey Moore distinguishes five customer groups: innovators, early adopters, early majority, late majority, and laggards. The innovators are technology enthusiasts; they typically buy and use your product as soon as it is launched. The early adopters are visionaries who can see how the product benefits them. Together, they make up the early market. Innovators and early adopters might accept a poor customer experience. They might not be put off if the product is hard to install, use, or update, and they might not expect that issues are quickly resolved by the support team. But that's not the case for the early majority.

29 Downes and Nunes (2013) suggest that certain disruptions only have two customer groups: *trial users* and *the vast majority*. The former roughly correspond to the early market and the latter to the mainstream market.

30 An *architecture refactoring* is a larger refactoring exercise that addresses not only individual classes and methods, but also the overall structure of a software product. While class-level refactoring should be part of the normal development work, architecture refactoring often requires a dedicated effort, such as a whole sprint or even an entire release.

31 SAFe stands for Scaled Agile Framework. It is one of several agile scaling frameworks available at the time of writing. See https://www.scaledagileframework.com for more information.

32 Moon (2005) argues that the product life cycle does not have to be linear, and that rejuvenating the product can be a great option.

33 https://www.statista.com/chart/7699/lp-sales-in-the-united-states/

34 For more advice on resolving collaboration issues and establishing effective connections with the stakeholders, see Pichler (2020).

35 A product focused on a specific segment can help establish a strong brand. Ries and Trout (1994) argue that a brand becomes stronger when you narrow the focus.

36 To learn more about the two techniques and how you can apply them, please refer to the *Product Strategy Validation* chapter.

37 Gray (2017) suggests also stating what someone hears and thinks. If that's something you can *confidently* state, then that's great. Just add new fields to your empathy map. Be careful, though, to stay clear of guesswork.

38 In some ways, a consumption chain map is similar to a user journey map. But unlike the latter, it does not use personas, which you might not have created at this stage. Additionally, it is more holistic: It describes the entire experience in one picture. These differences make the consumption chain map better suited to discovering needs in my view.

39 Ulwick (2016, Loc 759) calls this need *core functional job*, which the author views as "the anchor around which all other needs are defined."

40 Alan Cooper pioneered the use of personas in software development; see Cooper (1999).

41 This sample user story applies a popular template, which was originally developed by Rachel Davies at Connextra, see https://en.wikipedia.org/wiki/User_story.

42 Note that the Kano model offers two additional categories, *indifferent* and *reverse* features. As their name suggests, indifferent features don't influence customer satisfaction—people are indifferent towards them. Reverse features cause dissatisfaction; they can prevent people from using the product.

43 With the release of the iPhone 5S and 5C in 2013, Apple deviated from its original one-model strategy.

44 Other templates are also available to capture the product strategy, of course, including the *lean canvas* (Maurya 2012) and the *value proposition canvas* (Osterwalder and Pigneur 2014). Choose the one that works best for you, and experiment with different templates if you are in doubt.

45 I describe techniques to select the right stand-out features in the chapter *Strategy Development* and I share tips for creating a backlog in the chapter *Product Roadmap* Development.

46 The four sections were inspired by the sections on the business model canvas. The canvas has nine sections: customer segments and relationships, channels, value propositions, revenue streams, cost structure, key activities and resources, and key partners. Note that if you use the standard product vision board in combination with the business model canvas, there will be some overlap: The section *customer segments* is similar to *target group*, and *value propositions* overlaps with *needs* and *product*.

47 Rowe (2018) suggestions include following a design code of ethics, designing products that are good for humanity, and testing for abusability.

48 The criteria contained in the Definition of Done vary from product to product, but they usually include the requirements that the software can be executed on an official test server, that it has been tested, and that it has been documented. It may also state that the product increment can be released or shipped to (selected) users.
This usually includes the following requirements: the software can be executed; it has been tested; it has been documented; an

49 See: https://www.microsoft.com/en-gb/microsoft-365/visio/microsoft-visio-plans-and-pricing-compare-visio-options

50 Source: https://www.geekwire.com/2021/amazon-web-services-posts-record-13-5b-profits-2020-andy-jassys-aws-swan-song/, accessed on 14 December 2021.

51 IDE stands for *integrated development environment*. An IDE usually consists of a source code editor, a debugger, and additional tools that allow programmers to write, build, and test their code.

52 Sources: https://en.wikipedia.org/wiki/Amazon_Web_Services and https://www.geekwire.com/2021/amazon-web-services-posts-record-13-5b-profits-2020-andy-jassys-aws-swan-song/, both accessed on 14 December 2021.

53 For more information, see http://en.wikipedia.org/wiki/Browser_wars.

54 Before Microsoft released Word as part of Office for Windows in 1990, it was available as a stand-alone product for MS-DOS. See https://en.wikipedia.org/wiki/Microsoft_Word.

55 The product strategy validation I describe is based on the work of Eric Ries, especially his book *The Lean Startup* (Ries 2011). Note that I deviate from Ries' work by recommending that you identify risks instead of hypotheses. Risks and assumptions, however, are closely related: A risk is an uncertainty that may cause an undesired outcome if it is not addressed—much like an assumption that is not tested.

56 Drucker (1985, 163).

57 Source: http://en.wikipedia.org/wiki/Hackathon.

58 See Blank (2014).

59 I owe the last two tips to Blank (2014).

60 For a more detailed discussion of the techniques refer to Bland and Osterwalder (2019). The book nicely describes the methods including typical cost, setup time, evidence strength, and it offers additional techniques so you can pick and choose the ones that are right for you.

61 VoIP stands for *voice over IP*. The phone mentioned was a dedicated handset that used the Internet to make calls rather than a landline.

62 Flash was a once popular tool to create interactive multimedia content. It was discontinued in 2020.

63 The term spike was first used in Extreme Programming to refer to technical throwaway prototypes, see for example, http://www.extremeprogramming.org/rules/spike.html.

64 For more information on Kanban and working with a Kanban board, refer to Anderson (2010).

65 To determine the customer satisfaction, ask your customers how satisfied they are with the product using, for example, a survey with the following scale: *very satisfied*, *satisfied*, *neutral*, *unsatisfied*, or *very unsatisfied*. Then determine the number of people who are very satisfied and satisfied, divide this figure by the number of responses, and multiply it by 100. This results in the percentage of satisfied customers, which is also referred to as *customer satisfaction score* (CSAT).

66 Norton and Kaplan (1996) also distinguish four perspectives. But they use the terms *internal business processes* and *learning and growth* instead of *product and process* and *people*, respectively.

67 Sustainable pace is an important agile principle. It means that people follow a constant, healthy pace and are not overworked, certainly not for a longer period or on a regular basis. The Agile Manifesto for Software Development defines sustainable pace in the following way: "The sponsors, developers, and users should be able to maintain a constant pace indefinitely." See http://agilemanifesto.org/principles.html.

68 For more advice on collaborative decision-making see Pichler (2020).

69 For more advice on effectively collaborating with the stakeholders, see Pichler (2020).

70 The term *feature* is unfortunately rather ambiguous. As mentioned before, I view a feature as a product capability, as a large piece of functionality that is bigger than an epic (which is a large user story). I therefore derive epics from features (and user stories from epics).

71 While it's necessary to regularly review and update product roadmap, changes that happen too frequently can undermine the trust of the stakeholders and dev teams in the plan and might even render it useless. As a rule of thumb, a product roadmap should be stable for at least a month, preferably for two to three months.

72 See the section *Engage the Stakeholders and Development Teams* in the chapter *Product Strategy* Foundations for more information on how to determine the players including the key stakeholders.

73 I explain in more detail how the product roadmap helps you determine the right backlog items in the next chapter. For more information on the product backlog, see Pichler (2010).

74 I use the term *major release* to refer to a version of your digital product that introduces a noticeable change, for instance, by adding or optimising functionality or enhancing the user experience, and it typically results in a new product version—think of Windows 11 or iOS 15, for example.

75 See the section *Choose the Right Key Performance Indicators (KPIs) for Your Product* in the chapter *Product Strategy* Reviews for more information on KPIs.

76 A good resource to learn more about cost of delay and CD3 is Black Swan Framing, https://blackswanfarming.com

77 The acronym stands for *Reach*, *Impact*, *Confidence*, *Effort*—not to be confused with the injury treatment method also called RICE (Rest, Ice, Compression, Elevation).

78 The "now-next-later" time frames on product roadmaps were originally suggested by Janna Bastow, see https://vimeo.com/100642934.

79 Note that using this approach does not prevent you from continuously deploying smaller improvements and bug fixes.

80 See http://en.wikipedia.org/wiki/Timeline_of_web_browsers and http://en.wikipedia.org/wiki/Usage_share_of_web_browsers.

81 As mentioned before, asking people to do extra can reduce creativity, motivation, and productivity drop; people's wellbeing and health can suffer. Consequently, individuals might leave the team or even the company. Cutting software quality and creating technical debt makes it harder to update and change the product in the future.

82 Traditionally, the Iron Triangle looks at scope, time, and budget. To optimise it for goal-oriented roadmaps, I have replaced scope with product goal in Figure 27.

83 As I discussed in the chapter *Product Strategy* Reviews, I recommend adding software quality and sustainable pace to your KPIs. This will help you ensure that the product and development teams stay healthy. What's more, an agile framework like Scrum locks quality in the Definition of Done, and it empowers the development team members to decide how much work they can take on in a sprint. Another agile framework, Extreme Programming, offers a practice called *40-Hour Week* (Beck 2000). It states that people can't work overtime for two weeks in a row—which I've always considered a very practical application of sustainable pace.

84 Note that this approach is in line with the Scrum Guide 2020, which introduced the concept of a product goal to the Scrum framework. See https://scrumguides.org for more information.

85 I find it helpful to generally view existing requirements as a liability, not an asset. "A requirement simply describes product functionality that was thought to be necessary at some point in time. As markets and technologies change

and as the Scrum team gains more knowledge about how customer needs can best be met, requirements also change or become obsolete." (Pichler 2010, 52)

86 While the capabilities and the capacity of the development group often affect if creating a new product or feature is possible, I have seen organisations where marketing, legal, or another business group was the bottleneck, and hence determined the roadmap's feasibility.

87 Please see the section *Engage the Stakeholders and Development Teams* in the chapter *Product Strategy* Foundations for additional decision-making techniques.

88 See http://en.wikipedia.org/wiki/Murphy's_law.

89 It's been long known that a plan does not stay valid for long, especially in a changeable, volatile environment. Helmuth Karl Bernhard Graf von Moltke, a German Field Marshal who lived in the 19th century, famously stated that "no plan of operations extends with certainty beyond the first encounter with the enemy's main strength." In other words, no plan survives first contact with the enemy (Kenny 2016).

90 The chart is a tool commonly used in Scrum, see Cohn (2005) for a more detailed discussion of the tool. If you work with, say, a Kanban-based process, then you might want to employ an alternative like *cumulative flow diagram* instead (Anderson 2010). No matter what tool you choose, it should help you understand if the goal you are working on can be met on time and on budget.

91 This assumes that the development team can provide high-level estimates for all product backlog items that are required to meet the product goal.

92 Please see the section *Regularly Review and Update the Product Strategy* in the chapter *Product Strategy* Reviews for guidance on when and how to make changes to the product strategy.