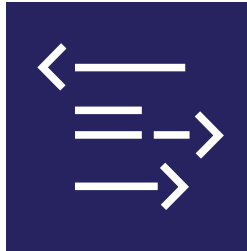
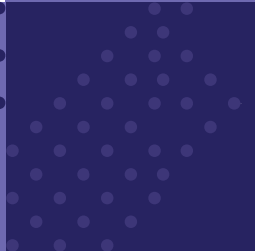
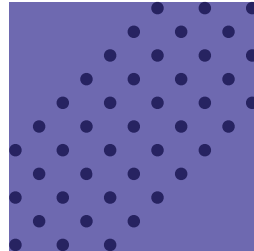
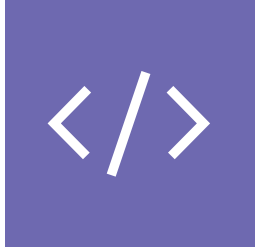


The Winning Product



*A Handbook for Launching
Successful Digital Products*

The Winning Product

A Handbook for Launching Successful Digital Products

99x
65, Walukarama Road,
Colombo 03,
Sri Lanka.

+94 114 72 11 94
hello@99x.io
www.99x.io



A Handbook for Launching Successful Digital Products

Contents

| | |
|---|-----------|
| Contributors | 10 |
| Acknowledgments | 11 |
| Preface | 12 |
| WHY THIS BOOK IS WRITTEN | 13 |
| Why are Digital Products special? | 14 |
| The odds of success and failure | 17 |
| Embracing the product mindset | 20 |
| Power Skills of a digital product team | 22 |
| The lifecycle phases of a digital product | 24 |
| PHASE 1 - EXPLORE THE LANDSCAPE | 30 |
| Evaluate the competition | 32 |
| Study the Mega-trends | 35 |
| Understand your users | 37 |
| PHASE 2 - FOCUS ON A PROBLEM | 40 |
| Verify the problem existence | 42 |
| Craft a business model | 44 |
| Discover core value drivers | 48 |
| Assess financial viability | 50 |
| PHASE 3 - IMMERSE IN THE SOLUTION | 52 |
| Prototype potential solutions | 54 |
| Establish the architecture blueprint | 56 |
| Formulate your competitive positioning | 58 |
| Craft a customer centric story | 61 |
| Identify your minimum viable product | 64 |
| Make the Go/No-Go decision | 67 |

| | |
|---|------------|
| PHASE 4 – PLAN YOUR JOURNEY | 70 |
| Derive ballpark estimates | 72 |
| Develop a product roadmap | 76 |
| Evolve high-fidelity prototypes | 79 |
| Establish a pricing strategy | 82 |
| Create a cohesive brand identity | 86 |
| Validate your product-market fit | 89 |
| Automate your delivery pipeline | 92 |
| Invest on a quality-driven culture | 95 |
| | |
| PHASE 5 - BUILD AND VALIDATE | 100 |
| Prioritize the product features | 102 |
| Establish a design system | 106 |
| Take control of code quality | 108 |
| Embrace test automation | 111 |
| Shift-left your quality gates | 114 |
| Establish a SecOps practice | 117 |
| Protect personal data | 119 |
| Setup your support processes | 123 |
| Delight customers at onboarding | 126 |
| | |
| PHASE 6 - OPTIMIZE THROUGH LEARNINGS | 128 |
| Leverage data for decision making | 130 |
| Develop your product portfolio | 134 |
| Maintain architectural consistency | 137 |
| Position for a rapid growth | 140 |
| Organize your lead funnel | 142 |
| Establish license management | 144 |
| Build a team brand | 147 |
| Manage cost escalation risks | 149 |

| | |
|---|------------|
| PHASE 7 - HARVEST YOUR RETURN | 152 |
| Nurture your leads | 154 |
| Embrace industry standards | 156 |
| Prepare for surge volumes | 158 |
| Tackle production issues promptly | 161 |
| Harness the power of inbound content | 164 |
| Embrace referrals and upsales | 167 |
| Prepare for contingencies | 169 |
| | |
| PHASE 8 - RETIRE VOLUNTARILY IN TIME | 172 |
| Reinvent your solution | 174 |
| Plan your migration journey | 177 |
| Deal with the residual data | 180 |
| So... what's next? | 182 |

Contributors

Contributors:

Amila De Silva
Arshad Nadheem
Ashan Fernando
Chamendri Silva
Chatura de Silva
Chrishan de Mel
Hasith Yaggahavita
Isuru Senadheera
Kamal Ratnayake
Manoj Fernando
Nawoda Herath
Pushpa Herath
Ruchira Prasad
Samudra Kanankearachchi
Sanara Premaratne
Sandrina Abeywardene
Vindya Gunawardena

Editorial Team:

Ashan Fernando
Amila De Silva
Hasith Yaggahavita
Chrishan de Mel

Design and layout:

Indika Gammudali

Reviewers:

Tarigo Product Management Training, UK
ChatGPT PLUS, Model GPT-4

Acknowledgments

“Nothing of significance was ever achieved by an individual acting alone. Look below the surface and you will find that all seemingly solo acts are really team efforts.”

– John C. Maxwell

As the editors, we appreciate the efforts of every contributor, reviewer and the 99x Winning Product community who made this publication possible.

This book is a testament to the product engineering DNA within 99x and our aspiration to build digital products that win in the marketplace. We count it a privilege to compile a record of it.

Editorial Team

thewinningproduct@99x.io

Preface

You are about to create your very own digital product. Or perhaps, you already have been working on a product for some time and even launched it to some customers. You've got the basics, you've read books, taken consultancy from experts, and feel you have what it takes.

Yet you know that there is a lot more ground to cover to become a winner. You are not alone. Many entrepreneurs have voyaged through these same waters.

Launching a successful digital product is not just about 'developing software'. The latter can be done by any skilled developer, but building products is done by one in hundreds. A digital product should not only satisfy the needs and wants of the end users but also meet the expectations of the product owner and investors. One must blend and balance multiple aspects such as technical excellence, user experience, market awareness, and customer success to launch a market-winning product.

While this is an intense task, it does not necessarily mean that you must figure your way out alone. Why not learn from the experiences of products launched in the past? What if you have a Map that tells you 'Where you are' and guides you on 'what's the most sensible thing to do next' – one step at a time?

Providing that guide is the focus of this book, and you will see more in the coming chapters. This book gathers hundreds of digital product development experiences and can serve as your strategic guide to decide where to focus as you navigate the journey of your digital product.

Why this book is written

| | |
|---|----|
| Why are Products special? | 14 |
| The odds of success and failure | 17 |
| Embracing the product mindset | 20 |
| Power Skills of a digital product team | 22 |
| The lifecycle phases of a digital product | 24 |

Why are Digital Products special?

"A lot of people have ideas, but there are few who decide to do something about them now."

- Nolan Bushnell

We build digital solutions either for internal use within an organization or to be sold as a packaged product. Have you realized that the two approaches have drastic differences? Well, in both cases, we gather requirements, develop, and deploy. In fact, when you develop a solution for internal use, these are the main activities you typically do. You may even hire external developers and tell them precisely what you expect from your internal system.

For example, a large bank might create bespoke software to handle its internal financial transactions and reporting. This software would be tailored to the bank's specific needs and not meant for use by other organizations.

Software for internal use is also called "bespoke software". So, if you hear the term 'bespoke', it's about software we build targeting a known userbase such as employees of a particular organization.

Comparatively, there is more sophistication required when building sellable digital software products that intend to compete in the market. Take, for instance, Slack, a communication and collaboration tool used by organizations worldwide. When developing Slack, the creators had to consider a much broader userbase who were personally not identifiable and ensure the product catered to a wide range of businesses with varying needs and sizes.

Unlike bespoke software, "commercial software" often begin without detailed data about a specific user base. Nevertheless, you must gain an idea of ultimate users by following user experience processes, such as defining personas and mapping user journeys.

Another example of this is Airbnb. In the early stages, the founders didn't have a clear understanding of their target audience. They spent time conducting user interviews, creating personas, and iterating on their product to meet the needs of hosts and guests, resulting in the global platform we know today.

Of course, there can be different interpretations to these words, but in this book to maintain consistency, the terms 'bespoke software' and 'digital products' will be used to differentiate these two types.

Why are Digital Products so Different?

Just to give you a glimpse of the sophistication, when launching a digital product, consider the types of activities you must do apart from solution development. You need an underlying business model, study the market, learn from the competition, find the channels to sell and distribute the product, make money, and the list goes on. Cool, so why not hire a few in business development and market research to deal with those things while you develop the product?

Unfortunately, that won't help much since the skillset required to launch a product is quite extensive. Sometimes, as an entrepreneur or the product owner you need to initially play most of these roles yourself. Unless you are funded in 10 digits, rob a bank or your surname is Gates, Musk or Bezos!

A real-world case study that demonstrates this is Uber. Uber's founders not only developed an innovative ride-hailing app but also conducted extensive market research, analyzed competitors like taxis and other transportation services, and developed a pricing model and marketing strategy to attract both drivers and passengers.

When it comes to software products, coding is just a fraction of the entire journey. You cannot overlook areas listed below that are specific to digital products.

1. Staying unique to stay ahead of market competition.
2. Convincing customers to get on-board and willingly pay.
3. Managing experience of a diverse user portfolio of multiple customers.
4. Offer high product agility and adaptability to multiple customer environments.
5. Flexibility to integrate with the brand identity of multiple customers.

Unfortunately, most amateur teams overlook the specialty of digital products and fall short of their true potential. To succeed in the digital product business, it is essential to know “when” and “how” to perform essential strategic activities to stay ahead of the game.

The focus of this book is to educate and guide you throughout this journey. Once you go through the remaining chapters, you will be confident in the winning product journey and learn from others who have gone before. How can you give your digital product the best chance of success? Let's find out in the next chapter.



While developing a bespoke solution is straightforward, launching a digital product demands broader insight, strategic versatility, and an entrepreneur's multifaceted role in ensuring the product's resonance in a competitive market.

The odds of success and failure

“An entrepreneur is someone who jumps off a cliff and builds a plane on the way down”

– Reid Hoffman



If your product is a sockeye salmon on its migratory journey, the odds are that there will be predatory bears waiting in the stream to finish it off. So, how can you give that salmon the best probability of making it to the destination?

First you must understand why a digital product is much more sophisticated than a bespoke application. While this was briefly explored in the previous chapter, for more clarity let us explore these differences in detail.

Understanding your user's needs and wants

Collecting requirements is invariably a complicated task, regardless of whether it's for bespoke software or a digital product. However, in bespoke software, organizations usually possess a considerably clear grasp of their needs and the organizational processes they wish to digitize. Identifying requirements through user interaction is simpler with bespoke software, as the requirements are tailored specifically for a single organization. For instance, consider developing bespoke software for an insurance company to digitize their claim management system. Developers can gain a clear

understanding of the requirements and the processes to digitize through direct interactions with the organizational users.

In contrast, imagine the complexity of gathering requirements for building a generic Insurance claim management product to be offered as a commercial-off-the-shelf (COTS) product. Determining the feature set and the business processes to implement for your target market can be a daunting task. It's unlikely that you will have access to all potential customers for your product. Even with a handful of potential customers participating in a pilot, they might not provide a comprehensive view of your product's future potential and the generic needs of your target market. Numerous assumptions must be made about the requirements. In digital products, users' needs must be cautiously balanced against the commercial viability across a wide range of potential market segments, which may result in challenging trade-offs.

Facing dynamics of market competition

While a digital product must compete and withstand the brunt of unknown market dynamics, an enterprise application is not expected to do so. For example, bespoke warehouse management software developed for a specific company does not have to face market competition. However, a warehouse management digital product must convey a competitive unique value proposition (UVP) and position itself attractively among hundreds of competing products.

Catering to unpredictable demands

The rollout for a bespoke enterprise application can be internally planned and agreed. You can have a roadmap, identify which user segment is onboarded first, and even predict the load on systems and support. However, there is no mercy in this when it comes to a digital product.

When you engage a prospective customer, a few seconds delay in functionality is enough for that prospect to look elsewhere. For example, an e-commerce platform may experience an unexpected surge in demand during a flash sale event. The production systems could struggle to cope with this heavy load, resulting in lost prospects.

Revenue and ROI pressure


A bespoke application usually operates under the financial protection of a parent organization. While there may be revenue expectations, the runway is usually longer for the application to safely take flight. The enterprise also has deeper pockets to tinker with the application until they get it right.

In contrast for digital products, it is a question of survival from day one. While managing your cashflows, you must battle to drive new sales, provide excellent customer service, reduce churn, improve and introduce new functionality.

Due to these factors, digital products have higher odds of failure and often end up either attempting to solve a non-critical problem, missing product-market fit or becoming poorly engineered. If you fall into any of these pitfalls, your message either does not attract your target market or you fail to secure enough customers.

For example, shared economy products like Uber and Airbnb face the challenge of stimulating both the supply-side and the demand-side simultaneously for the business model to work. This 'chicken and egg' situation usually means a significant investment which must be planned.

The odds of successfully launching a digital product are slim and call for greater agility and perseverance. In the next chapter we will explore the right mindset you must cultivate in your team to succeed.



Navigating the digital product landscape involves conquering the uncertainties of market dynamics, meeting unpredictable demands, and contending with revenue pressures, all requiring meticulous strategy, resilience, and a relentless pursuit of user value and market fit.

Embracing the product mindset

"Part of what made the Macintosh great was that the people working on it were musicians, poets, artists, zoologists, and historians. They also happened to be the best computer scientists in the world.

– Steve Jobs

“To succeed in any endeavor, you must have the right mindset.” This statement holds a deeper truth. Many unsuccessful products fail because the team lacks the determination to maintain momentum when facing challenges. Cultivating the right mindset, known as the “product mindset,” is a critical prerequisite for the team. This mindset can be further explained through the following three themes:

Continuous validation

The journey of your digital product does not stop at addressing your user’s needs. Digital products must evolve continuously, as market conditions and customer behavior are always changing. As a product team, you must adapt consistently to maximize value for your users. The agility to respond to market dynamics ensures a sustainable product position amidst competition. The key is to ‘get out of the building,’ gather customer feedback, and evolve as needed. Never stay complacent with your current success!

Product innovation

Successful products that boast a loyal customer base share a common ingredient: they go beyond a set of features and provide an innovative and compelling solution to the problem. Foster an environment that encourages team members to challenge the status quo and participate in the product journey. This involvement helps them feel a sense of accomplishment and the value of their contributions. Innovation can manifest in various forms: a technological breakthrough, a novel business model, improvements in efficiency and cost, or an unparalleled user experience.

Extreme ownership


Do you want to create a team of high-performing members who go the extra mile to deliver products customers adore? Then nurture the team to take extreme ownership of the product. The team should also love the product as much as users do. For example, consider how Steve Jobs attributed the success of the Mac to his team:

How can you achieve this? Begin by communicating the product vision and purpose. Ensure everyone believes in and aligns with the goal of solving users' problems. The product's purpose serves as the underlying adhesive that helps the team withstand challenging times.

Of course, it is not possible to cultivate such commitment overnight. It needs constant communication of the reason for your existence. Continuously reinforce the right behaviors and traits, conduct team retrospectives, gather feedback, and incorporate valid suggestions. As the product grows, so does the team.

Considering the challenges we've discussed thus far, launching a digital product is undoubtedly a daunting task. But what if there were a framework to organize all these activities? What if you knew the most crucial areas to excel in?

In the next few sections, you will be introduced to our Winning Product framework, which focuses on building digital products through focusing on four power skills and eight different lifecycle phases.



Cultivating a product mindset, embracing continuous validation, fostering innovation, and instilling extreme ownership are the cornerstones to navigating the intricate journey of digital product development successfully.

Power Skills of a digital product team

"The biggest risk is not taking any risk. In a world that's changing really quickly, the only strategy that is guaranteed to fail is not taking risks."

- Mark Zuckerberg

Technology is just one factor of a successful digital product. The formula for product success encompasses a much broader range of considerations, which can be categorized into the following four power skills.

Market Sense

Market factors are arguably the most critical ingredient for a product's success. Here, you examine how the product is valued by potential customers and, more importantly, how much they are willing to pay. Initially, analyzing megatrends helps you understand the general market direction. To gain a clear sense of the current competitive landscape, it's vital to conduct an in-depth analysis of other players' strengths and weaknesses.

For example, when Apple introduced the iPhone, they researched the market, analyzed trends in mobile technology, and identified a gap for a user-friendly, all-in-one device. This market sense allowed them to create a product that revolutionized the mobile phone industry.

Once preliminary analysis is complete, segment your target market and personalize product positioning. It's also beneficial to explore modern growth hacking techniques for rapid momentum and growth.

User experience

User experience focuses on deeply understanding users' expectations, experiences, and behavior. Easy-to-use products that empathize with users will consistently outperform competitors.

Identifying user personas is an excellent initial step. The more you know about your users, the more you can optimize the experience for greater stickiness. Once personas are identified, determine their journeys within the application and validate through prototyping. Analyzing users' pains and gains enables you to understand optimal user journeys. Once your product is in the hands of your users, there are some great tools to capture real behavior patterns. Continuous analysis of such information helps you make data-driven decisions.

Customer success

Customer success ensures your customers achieve their business goals by using your product. Primarily, your product should have a smooth onboarding process, engaging features and efficient support.

You might wonder, isn't it good enough to use a support system that captures customer feedback, requests, and support tickets? While reactive support is essential, delivering features and fixes through proactive understanding of user pain points can make a real difference.

In digital products, customer success ensures customers derive maximum value while using your product, translating to cross-selling, reduced churn, and maximized lifetime value in the long term.

Technical excellence

Technical excellence involves aligning architecture, development, and delivery practices with business priorities. However, teams may lose focus due to shifting priorities. For example, it's inevitable to build technical debt—a backlog of technical 'to-do' activities postponed for later as development evolves. A proper process must be in place to revisit this throughout the development lifecycle. Apart from architecture and development, you need a reliable release process, requirement management process, and quality assurance process. A clear product roadmap is essential, with prioritization based on the value and effort of each feature.

This section offers an overview of what's to come in later chapters. Each strategic activity that follows is categorized into one of the four power skills mentioned above. These activities are not one-time events but interact and iterate throughout the development process as the product evolves. Stay tuned—this journey is about to become real.



Success in digital product development hinges on four power skills: an intuitive market sense, unparalleled user experience, unwavering focus on customer success, and unwavering technological excellence.

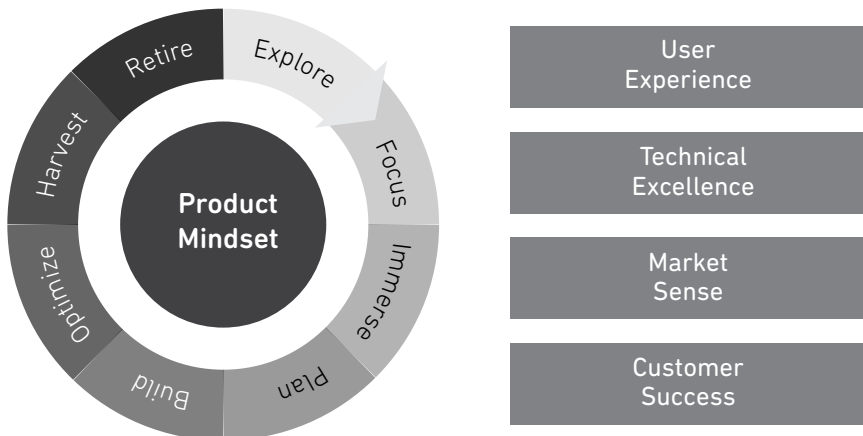
The lifecycle phases of a digital product

"Any product that needs a manual to work is broken."

- Elon Musk

All digital products have a life cycle. The Product Life Cycle (PLC) is the sequence of phases that a new or redeveloped product goes through. Understanding how to manage the product life cycle at each phase is crucial for steering your product towards success.

This book divides the product's lifecycle into eight phases. These phases are not necessarily sequential but iterative at the whole product or module level. Every strategic activity you need to perform under these phases is associated with one of the power skills we learned in the previous chapter.

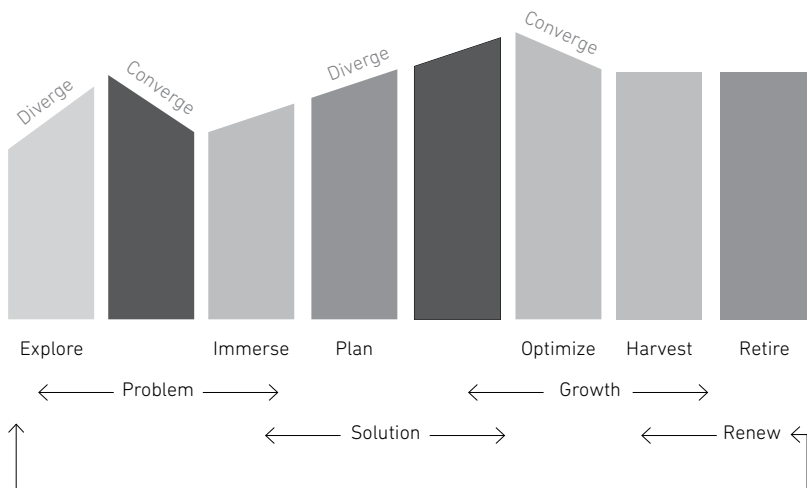


The Eight Phases of the Product Lifecycle and the Four Power Skills

As you navigate through the phases of the product lifecycle, you are answering four critical questions:

1. What problem should we solve?
2. How should the solution be designed?
3. How do we grow our product?
4. When and how should we retire the product?

In answering these questions, your team should embrace a specific way of thinking to achieve optimal results. These thinking patterns are known as Divergent Thinking and Convergent Thinking. Divergent Thinking involves exploring a wide range of options and ideas, while Convergent Thinking involves focusing on the most promising ideas and refining them into a workable solution. The figure below illustrates the eight phases of the product lifecycle and the corresponding thinking pattern your team must embrace.



Explore

This is the initial phase where you conceive the idea to solve a problem. It's crucial at this phase to take a holistic view of the problem domain and broadly explore the space around it. Investigate different possibilities before committing to a focused problem statement or solution approach.

Focus

During the Explore phase, you typically generate a vast number of ideas and value propositions your product could offer. The success of a product is not directly related to the number of features or the range of market segments. A smaller set of cohesive features is usually more effective than a large, incompatible set. Focus is the phase to narrow down attention to the most critical aspects of the problem. You need to start locking down on a problem area to solve at this phase.

Immerse

Once the product focus (product niche) is identified, it's time to dive into the details of the problem to find a solution. During this phase, consider required features, associated costs, and timelines. A lot of 'what if' analysis takes place, and the base artifacts required for detailed planning are generated here.

Plan

With a clear understanding of the required solution from the Immersion phase, create initial implementation plans. These plans are live documents that are updated throughout the lifecycle. This phase is not about creating a comprehensive plan upfront, but rather establishing agile base documents and processes.

Build

This is the phase where most of the core features get developed. You have the highest flexibility to change, modify, or pivot the solution. Once the product passes this phase, changes become costly due to technical complexities and stakeholder involvement. Validate all assumptions (technical, requirements, business) made during previous phases through MVPs, demos, user testing, and architecture proof of concepts (POCs). Consider onboarding pilot customers as testers to verify your value proposition.

Optimize

By this phase, all major assumptions should be validated and proven useful to customers. Your digital product's unique market positioning must be clear, and customers' willingness to pay should be confirmed. Most core features are in place and accepted by pilot customers. During optimization, further fine-tune the product to maximize its potential within your

market segment. Carry out product updates required for adoption by more customers. Strive for an optimal balance between product quality, stability, visual appearance, onboarding procedures, and cost optimization.

Harvest

At the harvest phase, your product begins to yield a significant return on investment. During this phase, it is crucial to maintain happy and loyal customers, upsell, and acquire referrals to grow the business. Stable operational processes are essential for addressing increasing customer concerns and inquiries. Enhance your sales and marketing strategies to capture a larger market share, as your value proposition is now well-established.

While your product may be successful, it is not immune to obsolescence. Continuously invest time and resources to stay ahead of the competition. Emphasize a research and market-driven approach to ensure that new features or improvements are relevant and necessary while considering associated costs, risks, and potential technical debt. This will help you avoid introducing new issues and maintain your product's stability and reliability.

Retire

No matter how well-built or maintained a product is, there comes an optimal time for retirement. This can be triggered by a significant shift in technology or user behavior. Product companies must remain vigilant to identify this timing and disrupt their own business model before a competitor does. At this phase, reinventing the technology platform, business model, and user experience is essential to maintain your competitive advantage.

When planning for the next-generation product, ensure minimal disruption to existing customers. Offer a seamless migration path and incentives for transitioning. Most product companies provide both versions side-by-side during this phase. However, operating two major versions simultaneously can be costly, leading some companies to adopt a more incremental approach. Regardless of the method chosen, you must aim to migrate all customers within an acceptable time frame.

With this understanding in place, let's examine the first phase of your product journey. Where should you start? What information do you need? Who are your competitors, and what strategies are they employing? Explore your problem domain and embark on your product's journey.



Every digital product undergoes a lifecycle from exploration to retirement; navigating each phase with clarity and purpose ensures lasting success and adaptability in an ever-evolving market.

Phase 1

Explore the landscape

| | |
|--------------------------|----|
| Evaluate the competition | 32 |
| Study the Mega-trends | 35 |
| Understand your users | 37 |

Empathy is a vital aspect of the initial “Explore” phase of the product lifecycle, where the founder conceives an idea to address a problem. It’s crucial to put yourself in the shoes of potential users and adopt a user-centered approach to problem-solving. Delving into user needs and desires can reveal genuine challenges and pain points. This exploratory phase encompasses activities such as conducting market research, assessing mega-trends, and investigating competitive offerings. Ultimately, taking an empathetic and exploratory approach to the problem can help you create a novel solution that genuinely resonates with your users and addresses their problems in a meaningful way.

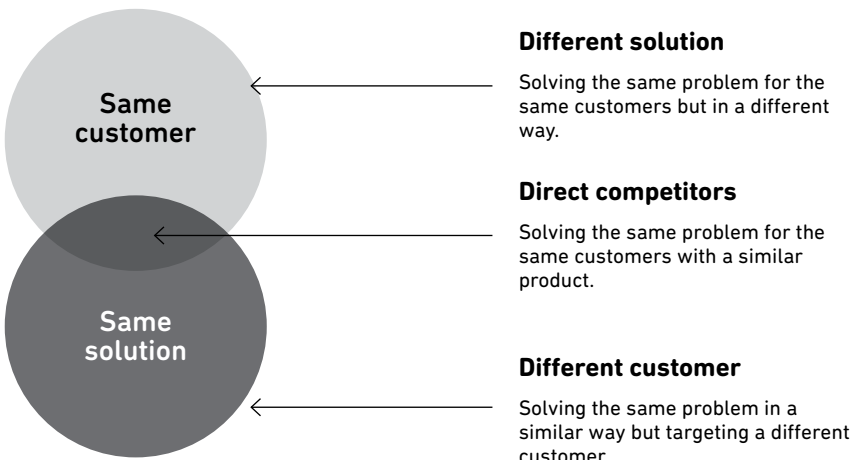
Evaluate the competition

"The art of life is a constant readjustment to our surroundings."

- Kakuzo Okakura

As an enthusiastic entrepreneur, you have yet to establish yourself in the industry. Studying existing competitors and their approach to solving the problem you identified can provide valuable insights and broaden your understanding of the problem space. Competitor analysis is one of the most cost-effective ways to enhance your knowledge.

New ideas always seem fresh, but you will likely find competitors in your space. If you don't, this may indicate a non-existent or financially unviable market. Regardless of your idea's innovation, it's essential to analyze your competition to determine the optimal positioning for your new product offering. Generally, you will find three types of competitors as shown in the diagram below.



Your analysis should not be limited to direct competitors; study all three types. Rather than a feature-to-feature comparison, evaluate competitor offerings from the target customer's perspective. To optimally position your product, consider the following points to ensure you identify a financially viable market segment where customers are underserved:

- Competitor unique positioning and price point
- Weaknesses in competitor offerings you can capitalize on
- User segments that are not covered by your competition
- Growth rate of your competitors and fast-growing markets


During the analysis, document your findings in a competitor analysis matrix to organize your data meaningfully. Below are some factors to consider for comparison:

| | Competitor 1 | ... | Competitor n | Your product |
|------------------|--------------|-----|--------------|--------------|
| Differentiator | | | | |
| Target audience | | | | |
| Market share | | | | |
| Revenue growth | | | | |
| Pricing model | | | | |
| Sales strategy | | | | |
| Product features | | | | |
| Strengths | | | | |
| Weaknesses | | | | |

You should spot and study both **Sharks** and **Underdogs** during your competitor study. Sharks are powerful players with deep pockets, poised to dominate the market. They have strong financial backing and the resources to pursue aggressive strategies. In contrast, Underdogs are smaller players operating under-the-radar. Their agility allows them to challenge the Sharks by offering differentiated solutions to niche markets. For example, when looking at the smartphone industry in the early 2020s, Apple and Samsung are considered Sharks while OnePlus and Xiaomi can be seen as Underdogs that have managed to make a significant impact in the market with innovative offerings and competitive pricing. Never underestimate underdogs in your market!

Once you understand your competition, focus on finding market gaps and using your strengths to address these opportunities creatively. Optimal market positioning is a crucial driver of success.

Apart from competitors, you must also be aware of shifts in your market. Macro-level changes can impact the entire industry, triggered by one-off events or gradual movements over time. These shifts will have significant long-term consequences. In the next chapter, we will learn how to respond to and capitalize on these mega-trends.



To launch a standout digital product, one must delve deeply into competitor analysis, identifying market gaps while respecting both the Sharks' dominance and the agility of the Underdogs.

Study the Mega-trends

"The ability to learn faster than your competitors may be the only sustainable competitive advantage."

- Arie de Geus

Mega-trends are profound shifts affecting your industry at a macro level. Mega-trends influence the behavioral or attitudinal changes of users over time. Conducting a mega-trend analysis is essential to stay relevant in the long term.

The most challenging mega-trends are often the fuzzy ones, which influence different markets with varying depths and speeds and are difficult to predict. Although this is a common challenge for any product domain, fostering agility to quickly adapt to such trends can be rewarding.

Mega-trend analysis allows your product to develop proactive long-term strategies and align them with growth plans. Below are a few steps that can help you perform a mega-trend analysis:

1. Identify the driving factors

First, identify the factors affecting your industry. The factors mentioned below provide a starting point for understanding the potential driving forces in your specific sector.

- Aging population
- Shifts in global economic power
- Connected consumers and businesses
- Climate change and sustainability
- Geopolitical tensions
- Availability of vast amounts of data
- Emergence of advanced AI systems

2. Listing of trends

Next, list the trends that impact your industry and assess their potential by brainstorming or speaking with domain experts. For instance, the aging population has spurred tech innovations in elderly care for both care homes and private residences. Additionally, the COVID pandemic forced healthcare businesses to adopt telemedicine, causing significant disruption.


3. Address the mega-trends

Not every trend is a mega-trend. Focus on identifying the mega-trends most relevant to your product offering and user base. Some of these trends can be deceiving and short-lived, so it's important to establish a proper process for addressing the most impactful ones. The specific significance to your product may depend on the severity and longevity of the impact.

Keep in mind that many products have failed to observe mega-trends and paid the price. Effectively harnessing a significant mega-trend may provide a clear window into the future and position you as a market disruptor.

A noteworthy example of a company that failed to recognize and adapt to a mega-trend is Blockbuster. Once a dominant player in the movie rental industry, Blockbuster relied on its physical stores to provide customers with movies and entertainment. As internet and streaming technologies evolved, new market entrants like Netflix identified the potential of delivering content to consumers through online platforms. Blockbuster was slow to respond and adapt to the new business model. This allowed Netflix and other streaming platforms to capture a significant market share.

Having understood the Macro market perspective, it is important to now shift our attention to the people who will ultimately use your product – users. Gaining a deep understanding of your users is essential for creating a product that addresses their needs, solves their problems, and provides them with a satisfying experience. Let's now focus on understanding your users.



Anticipating and embracing mega-trends secures your product's future, ensuring it thrives in a dynamically evolving market landscape.

Understand your users

"You've got to start with the customer experience and work back toward the technology, not the other way around."

- Steve Jobs

Software products are designed to cater to users' needs. Therefore, understanding who your users are, how they interact with your product, and how much value each feature provides them is crucial. Imagine investing time and money in developing features only to discover that just a fraction of your users' needs them. To avoid this, start by identifying different types of users and what matters to them through user segmentation.

User Segmentation


User segmentation involves dividing your target users into distinct categories based on demographics, interests, needs, or location. By understanding these user groups, you can create personas and journey maps tailored to each segment.

Identifying User Personas

A user persona is a fictional representation of a specific user segment in your system. For example, "Mobile Mary" might represent all users who access your system via a mobile phone to consume content. By analyzing personas, you can identify the goals and frustrations of each user segment and better understand their needs. Given below is a sample persona chart.

Emily Johnson

| | |
|---|--|
| <p>Emily is an ambitious marketing professional with over eight years of experience. Starting as a marketing coordinator at a small startup, she rose through the ranks to become a highly respected Marketing Manager at a mid-sized technology company. In her free time, she enjoys exploring nature trails with her dog, Max, and pursuing her passion for photography.</p> | <p>Age: 32 Gender: Female Occupation: Marketing Manager Education: Bachelor's degree Location: Seattle, Washington, USA</p> |
| <p>Goals: Personal and professional growth Thrives on challenges Continuous learning and skill improvement Efficiency and effectiveness Making a positive impact on the world</p> | <p>Frustrations: Stagnation in her career Ineffective marketing strategies Lack of data-driven decision-making Difficulty achieving work-life balance</p> |
| <p>Personality: Analytical thinker with a creative streak, logical mindset, detail-oriented and appreciates well-designed experiences. Empathetic and attentive listener.</p> | |



The best way to capture this information is to observe and interview potential users. You can follow several best practices during this process.

- Research your target audience to identify their demographics, concerns, and current solutions.
- Identify the most common attributes shared by the selected user segment.
- Create a persona that captures demographics, pain points, tasks they want to accomplish, personality, and mental state.
- Assign a unique name to each persona for easy identification.

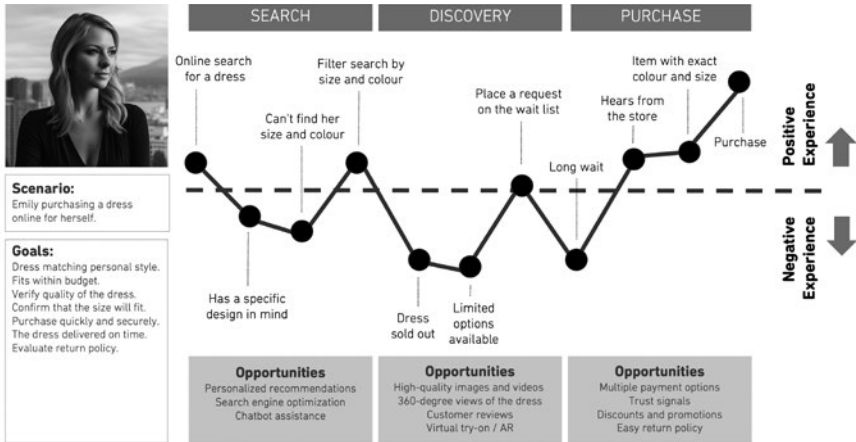
This process allows you to empathize with your users and see things from their perspective. The next step is to optimize their engagement with your product by identifying user journeys.

Identifying User Journeys

Isn't it just the requirements and use cases we are talking about here? No, there's more to it. User journeys go beyond traditional requirements and use cases. They visualize users' experiences as they interact with your application to achieve their goals. By mapping these interactions, you can gain insight into users' emotional experiences at each stage and identify opportunities for improvement.

Building a Journey Map itself is a journey. Customer narratives are used to plot their experience over time, map their activities and assess their mental state. The following steps will give you an idea of how to create your own Journey Map.

- Select a persona and a goal.
- Identify the user interactions required to achieve this goal.
- List the pain points and expected rewards at each stage.
- Determine the emotions users might feel throughout the journey.
- Identify opportunities for improvement to enhance user experience.



In this Explore phase, we took a broad look at the possibilities. However, it won't take that long for you to realize that the total problem scope is too big to make effective decisions. As we move into the Focus phase, let's converge our efforts to address a specific problem from all potential avenues. This approach ensures that you cater to a focused set of needs and wants, resulting in a stickier product.

Truly understanding your users through careful segmentation and persona development enables the creation of products that resonate deeply, meet needs effectively, and deliver unparalleled user experiences.

Phase 2

Focus on a problem

| | |
|------------------------------|----|
| Verify the problem existence | 42 |
| Craft a business model | 44 |
| Discover core value drivers | 48 |
| Assess financial viability | 50 |

In the previous Explore phase, you might have been buzzing with creativity, coming up with diversified regarding the problem you want to tackle. As you step into the Focus phase, it's time to zoom in, and pinpoint the specific problem space you're eager to focus on. This essential phase will nudge you to confront the validity of your problem, fine-tune your business model, refine your value proposition, and assess the financial viability within your target market segment.

Verify the problem existence

"Value is what people are willing to pay for it."

- John Naisbitt

Before investing time and resources into building a product, it's crucial to validate that the problem you're solving is worth solving. As research conducted in 2019 across 300 global startups revealed, 70% of new ventures failed due to their tech products not addressing a real customer's need or there being no market for what was built. To avoid becoming part of that statistic, it's essential to determine early on if your product idea is worth developing. There are two critical questions we need to answer here.

1. Is the problem you are trying to solve, worth solving? I.e., Is the problem you are trying to solve big enough so that people are willing to pay for a solution?
2. Is your solution good enough to be recognized by customers as a solution to the defined problem? I.e., Will the customer pay for your solution?

Problem-Solution Fit is achieved by carefully refining both the problem to be solved and the proposed solution, to the point where there is a high degree of confidence that customers will be willing to pay for it.

To test Problem-Solution Fit, it's important to interact with potential customers and gather feedback. Getting feedback based on an MVP (Minimal Viable Product - which will be discussed later in the book) is a great way to learn about the customer pain points, an MVP can be a zero-code mockup which is less expensive but highly effective. Getting out of the building and meeting your prospects is essential to understanding their pains and gains. This is essential to identify if your product solves a real-world business problem. Here are some common questions to ask your potential customers to help determine if your product idea is worth pursuing:

- How do customers rate your core value proposition? How much are they willing to pay for it?
- How many customers would be willing to come on board immediately if a working solution is available? Are they willing to sign a letter of intent?

- How does your value proposition compare to the available competition? Are your customers willing to switch from your competitor to your product?
- What are the high priority features your customers would be willing to pay a premium?
- Does your solution leverage the right megatrends your customers are aspiring to onboard?

It's important to realize that technical brilliance or state-of-the-art technology alone does not guarantee success. Many of the best technical ideas fail if there isn't sufficient demand from customers. A good example is Google Glass. While the technology was impressive, it ultimately failed because there wasn't enough demand from consumers, and it lacked a competitive price point. In 2015, Google ended the project due to poor interest from the market.

Therefore, the main objective of validating the problem is to know if the idea addresses a real-world business problem and has sufficient user-base willing to pay for the solution. If your validation results aren't favorable to your offering, be willing to pivot and try something different. A good example of such a pivot is Slack, a popular team communication platform, started as an online multiplayer game called Glitch. When the game failed to gain traction, the team recognized that their internal communication tool, developed for game development collaboration, had potential as a standalone product. They pivoted and rebranded, launching Slack in 2013. The platform quickly gained popularity, transforming team communication in workplaces worldwide. Therefore, continue to conduct user research until you find a problem worth solving. Once you have identified a problem worth solving, you can move on to creating the right business model around it, which we'll explore in the next chapter.

Validating the existence of a problem and its worthiness to solve is the cornerstone to building a successful product; it's the compass guiding you to create solutions that truly resonate with customer needs.

Craft a business model

Startups don't fail because they lack a product; they fail because they lack customers and a profitable business model.

- Steve Blank










Crafting a business model is more than just finding out how to sell your product. The process involves mapping out your financial plans to your value proposition—a promise you make to your stakeholders. Keep in mind, all these elements need to work together well for your business model to be successful.

Take PayPal, for example. It started with a crystal-clear value proposition for a small, niche group of power users. As Peter Thiel stated in his book, Zero to One:

The most successful companies make the core progression, dominating a specific niche and then scaling to adjacent markets a part of their founding narrative.

- Peter Thiel

The Business Model Canvas is a one-page document that summarizes your business, as shown below:

| | | | | |
|--|---|---|---|--|
| Key partners  | Key activities  | Value propositions  | Customer relationships  | Customer segments  |
| | Key resources  | | Channels  | |
| Cost structure  | | Revenue streams  | | |

Every business operates with multiple touchpoints and stakeholders, resulting in a complex system. A well-crafted business model should convey how a company plans to operate, connect with stakeholders, create value, and rake in the cash. This approach provides a comprehensive overview of how the business's market offering links with its internal workings. Examining all these aspects in one place helps you spot connections you might otherwise miss.


To create a Business Model Canvas, fill in each cell in the order given below, starting with the value proposition and customer segments. Each cell should address critical questions like:

- 1. Value Proposition:** The value proposition highlights the unique features and benefits that set your product or service apart from competitors. It answers the question of why customers should choose your offering over others. This includes the problems it solves, the needs it meets, and any differentiators that give you a competitive edge.
- 2. Customer Segments:** Customer segments are the different groups of people or organizations your product aims to serve. Each segment may have distinct needs, behaviors, and preferences. Understanding these segments helps you tailor your value proposition, marketing efforts, and product development to better address their specific requirements.
- 3. Channels:** Channels are the various means through which you reach, communicate with, and deliver your product or service to customers. These can include online platforms like websites, mobile apps, social media, as well as offline avenues like brick-and-mortar stores, events, and partnerships with resellers or distributors.
- 4. Customer Relationships:** Customer relationships refer to the strategies and processes you put in place to acquire, retain, and grow your customer base. This can include activities like marketing campaigns, personalized customer support, loyalty programs, and community engagement initiatives aimed at nurturing long-term relationships with your customers.

5. **Revenue Streams:** Revenue streams are the different ways your product generates income. This can include sales of products or services, subscription fees, advertising revenue, licensing fees, or even commission from affiliate partnerships. Identifying and optimizing multiple revenue streams can help improve your financial stability and growth.
6. **Key Resources:** Key resources are the assets and capabilities required to execute your business model effectively. These can include physical resources (e.g., facilities, equipment), human resources (e.g., skilled employees, leadership), intellectual resources (e.g., patents, know-how), and financial resources (e.g., funding, cash reserves).
7. **Key Partners:** Key partners are external organizations or individuals that contribute to your product's success. These can include suppliers, distributors, technology partners, strategic alliances, or even industry associations. Collaborating with key partners can help you access essential resources, expand your market reach, and improve your product offerings.
8. **Key Activities:** Key activities are the critical tasks and processes necessary for your product to deliver its value proposition, reach customers, and generate revenue. These can include product development, manufacturing, marketing, sales, customer support, and other operations essential to your business model's success.
9. **Cost Structure:** The cost structure outlines the various expenses incurred in running your product business, including both fixed and variable costs. This can include costs related to production, marketing, staffing, rent, utilities, research and development, and more. Understanding your cost structure helps you optimize expenses, improve profitability, and make informed decisions about pricing and resource allocation.

The Business Model Canvas offers a chance to visualize and evaluate your product idea or concept. It's a living document that you'll need to revisit and update regularly with relevant and accurate information.

Keep in mind, unless you've tested your business model with potential customers first, your business plan is nothing more than a work of fiction. The business model too, evolves through continuous improvement. The more you test your model, the stronger it'll be.



Crafting a robust business model is a foundational step; it's about intertwining your value proposition with financial plans and continuously refining them based on customer feedback, ensuring the sustained success and evolution of your product.

Discover core value drivers

"Fall in love with the problem, not the solution, and the rest will follow."

- Uri Levine

In the early phases of your product, it's essential to understand the factors that drive its value. By knowing this, you can focus your efforts on activities that enhance your product's worth. Let's explore both internal and external factors.

Internal factors

Every product is unique in how it derives its value. For some, it could be the number of users, or the data collected from them, while for others, it's the growth in direct revenue and profitability. It could even be the community goodwill and brand-building your product delivers.

For example, Facebook initially focused on growing its user base and encouraging active engagement, which later became crucial for targeting advertisements, its primary revenue source.

To identify what really matters to your product's value, below is a list of drivers to consider together with examples.

Revenue: Apple focuses on generating revenue through its products, including iPhones, iPads, Macs, and its services like Apple Music and iCloud. Their value is directly tied to the sales and profit margins.

Number of users: Facebook's value comes from its user base, which drives advertising revenue. The company focuses on user growth and engagement, providing targeted advertising opportunities to Facebook's clients.

User behavior data: Google's value is in its vast amounts of user behavior data. This data enables them to deliver highly targeted advertisements to individuals.

Customer segments: Amazon's value comes from its ability to cater to diverse customer segments, from individual shoppers to businesses. They offer a range of products and services, including retail, cloud services, and subscriptions tailored to different customer needs.

Market share: Microsoft's value is derived from its dominant market share in the enterprise markets. Its Operating System and the Office suite have become indispensable tools.


Partnerships: Salesforce's value is in its strategic partnerships with partners, such as system integrators, Independent Software Vendors, and resellers. These partnerships enable Salesforce to provide comprehensive solutions across various industries.

External factors

Various external factors also drive your product's valuation. For instance, building a product that emphasizes "Security" will have more credibility based on its origin. If the business operates in a country that doesn't value security and privacy, its value could suffer compared to the competition.

External factors like legislation, regulatory changes, and global phenomena can shift a value proposition from "Nice to have" to "Must-have." For example, Zoom's product valuation skyrocketed almost 10x within the first year of the pandemic.

Keep a close eye on both internal and external factors that drive your product valuation. Valuation is often connected to your business's financial viability too. Let's delve into that next.



Understanding and amplifying the internal and external value drivers unique to your product is pivotal for ensuring its growth, relevance, and long-term financial success.

Assess financial viability

“Sometimes there is a gap in the market because there’s no market in the gap”
- Irene Bejenke Walsh

You’ve had your eureka moment, and you’re wondering why no one else has thought of it yet. Perhaps you’ve even daydreamed about the success that could come your way. Irene Walsh’s quote is a reality check that could be a lifesaver. It will make you consider whether your brilliant idea has the potential to gain traction, generate revenue, and support a sustainable business. If you don’t have a financial background, now is a good time to familiarize yourself with the essentials of financial management.

The excitement of solving your users’ problems can sometimes overshadow financial considerations. So, don’t lose sight of budgeting your cash flows and profit margins. When starting a new product business, don’t underestimate the importance of financial planning and sustainability. You should have a clear plan for generating revenue, forecasting and managing expenses, and maintaining liquidity for operational costs. Focus on capital requirements, revenue projections, breakeven analysis, and cash flows while juggling everything else.

How do you assess financial viability?

- 1. Sales and revenue potential:** Ensure the market is large enough to generate the revenue needed for a scalable business. Identify the sales volumes you can achieve with your available resources. For example, Airbnb recognized the potential market for short-term property rentals and successfully tapped into it, generating significant revenue.
- 2. Cost of sales and expenses:** Factor in costs associated with lead generation, customer acquisition, product development, infrastructure, licensing, payroll, and more. For example, Uber had to account for driver incentives, marketing campaigns, and technology development in its financial planning. It had been a long ride for Uber to turn a profit on a net basis.

3. **Profitability:** Evaluate your gross profit and net profit margins. Products with higher gross profit margins can generate more profit as they scale, making them more attractive to investors. For example, SaaS companies like Slack typically aim for gross profit margins of around 80%.
4. **Cash flow analysis:** Ensure your upfront investment and cash inflows are positive. As Peter Drucker said, profit is secondary—cash flow is vital for survival. For example, Amazon focused on cash flow and reinvested profits into growing the business, even during periods of low net income.
5. **Breakeven Analysis:** Determine when you can recover your investment. For instance, if development costs are US\$100,000 per year and your product is priced at US\$10, you need to sell 10,000 licenses each year to break even. Ensure this aligns with your addressable market size and consider revisiting your pricing strategy or business model if necessary.
6. **Sources of funds:** Determine your initial capital, which could come from savings, support from friends or family, loans, or seed investors. Understand your financial runway based on available or accessible funds. For example, Tesla secured funding from Elon Musk and other investors to fund its long runway to profit.

Financial viability will easily make or break your product business and shouldn't be taken lightly. If finance isn't your strong suit, seek assistance. Regular support from someone who can assess your financial feasibility is crucial for your product's survival.

Ensuring your brilliant idea is also a financially viable one is crucial; rigorous financial planning, clear revenue strategies, and continual assessments are the lifelines for your product's sustainability and success.

Phase 3

Immerse in the solution

| | |
|--|----|
| Prototype potential solutions | 54 |
| Establish the architecture blueprint | 56 |
| Formulate your competitive positioning | 58 |
| Craft a customer centric story | 61 |
| Identify your minimum viable product | 64 |
| Make the Go/No-Go decision | 67 |

With your product focus (niche) identified, it's time to delve into the details of the problem and uncover the ideal solution. This phase involves empathizing with your users to create a solution that truly resonates with them. By using techniques like immersive empathy, you can develop solutions that form emotional connections with your users.

The goal of this phase is to outline the initial solution, covering aspects such as product features, technology, and market positioning. Embrace the process and be prepared to iterate as you learn more about your users and their needs.

Prototype potential solutions

Consider spending less time talking, and more time prototyping, especially if you're not very good at talking."

- Paul Buchheit

Now that you've identified a problem worth solving and established a financially viable business model, it's time to shift from the problem space to the solution space. Focus on the core features and flow of your solution, aiming to validate it as early as possible to give you room for pivoting if necessary. Prototypes enable you to test your ideas before investing in more expensive and time-consuming development phases.

There may be multiple ways your product can deliver the same functionality to end-users. Rapid prototyping helps you determine which alternative flows resonate best with them. Use these prototypes to gather user feedback and fine-tune the ideal problem-solution fit. Remember, speed is crucial—don't spend too much time perfecting each idea, as much of the work done in this phase might be discarded.

Fail fast. Fail cheap.

A great example of a SaaS product company that used rapid prototyping effectively is Airbnb. In their early days, the founders created several low-fidelity prototypes to test various booking flows, user interfaces, and features before settling on the final version that we see today. This process allowed them to iterate quickly and discover the most user-friendly and efficient solution.


Another example is Slack, the popular team communication platform. During its development, the team employed rapid prototyping to quickly create and test different user interfaces and functionality ideas, eventually leading to the intuitive and user-friendly product that millions of users rely on daily.

Prototyping also facilitates engagement with other stakeholders, such as developers, testers, UX designers, product owners, and even investors. As prototypes are visually engaging, they help your team identify pain points and opportunities from different perspectives and align your product's direction.

Rapid prototyping is cost-effective and boasts a high ROI. Effective prototyping demands collaboration and commitment from all stakeholders. Consider using paper mockups, flipcharts, or whiteboards for low-fidelity prototypes.

Avoid high-fidelity prototypes at this stage, as users may perceive them as finished products, hindering constructive feedback on the core solution. As you gain more confidence in your solution, you can later create a high-fidelity prototype, which is more interactive, polished, and often designed using digital tools. This will help you validate the aesthetic appeal of your product for your target audience.

Rapid prototyping serves as a valuable tool for validating your solution and pinpointing where to invest your efforts. It acts as a bridge between the business and engineering worlds. Business users provide insightful feedback and take more ownership when they see something tangible. For engineers, visual aids are crucial for understanding detailed requirements, and prototyping helps minimize rework in later phases, ultimately saving time and resources.



Venturing into the solution space, rapid prototyping is your compass, allowing you to validate core features swiftly and efficiently, enabling invaluable user feedback and facilitating a unified vision, steering the journey from idea to tangible product.

Establish the architecture blueprint

"The function of good software is to make the complex appear to be simple."
- Grady Booch

Creating an optimal product architecture is crucial for your success, in addition to the business-focused activities already discussed. Effective architectures define how software components are engineered and the interactions between them. Your architecture should be flexible enough to support the future growth of the product. Selecting the right technical approach while analyzing tradeoffs is essential for success. Your choice might depend on prioritized quality attributes like throughput, security, or cost aspects. It's important to make this decision rationally, based on your requirements and operational context.

Modern-day digital products use a variety of complex technical resources from different vendors and communities. However, business requirements should have the highest priority in governing architecture choices. Architecture is a living document and should be handled in an agile manner. Let's explore some concepts to help you create the optimal architecture for your product.

The architecture blueprint

The architecture blueprint visually conveys and justifies your product platform decisions. It should include functional and non-functional considerations and constraints of tech-stack, platform, deployment model, and DevOps processes. These can be viewed from different perspectives, such as conceptual, logical, or physical levels. For example, Netflix has a well-documented architecture blueprint that shows how they handle massive traffic loads and deliver seamless streaming experiences across devices. Their blueprint highlights their microservices-based approach and use of distributed databases. The team should make essential architectural decisions at this initial point and adhere to them going forward. Once the architecture blueprint is in place, it serves as a guideline for further development.

The architecture runway

You can develop a detailed design from the beginning, known as "Big Design Up Front." However, this approach can lead to overengineering and unnecessary complexity, making maintainability more difficult and costly.

Alternatively, if you keep the architecture too loose and agile without proper planning for potential growth, it may be insufficient to scale. Thus, the concept of an architecture runway is to find the right balance between these two extremes. Good architects tend to delay architecture decisions to extend the architecture runway; this practice is called 'deferred decisions.'

Quality Attributes

Once you understand this balance, the next part of the plan is to define the quality attributes. Quality attributes are technical and non-functional requirements significant for architecture. The priority you give to each one of these depends on the specific needs of your product. For instance, confidentiality is a high priority for a healthcare application, while performance would be the highest priority for a stock exchange system. Be aware of the possible trade-offs and conflicts that can arise from using different tactics you place to meet such quality attributes. For example, tactics used for high security may impede usability and performance.

Architecture Validation

The last part of a good architecture is validation. Good architects validate their architecture at the early phases. Validation usually comes in terms of assessing the limitations of the architecture against the expectations of the quality attributes.

In conclusion, creating an effective product architecture is essential for success, and choosing the right technical approach requires analyzing tradeoffs and prioritizing business requirements. An optimal product architecture that is flexible enough to support future growth and serves the essential architectural quality attributes can make a significant difference in the product's success.



Crafting a balanced, agile, and living architecture is a pivotal step in product creation; it's the beacon that guides cohesive development and innovation, allowing your vision to adapt, scale, and realize its full potential.

Formulate your competitive positioning

"People don't buy what you do; they buy why you do it."

- Simon Sinek

The competitive positioning statement (aka. Unique value proposition) is an internal guideline used to align your company when making decisions related to your product's strategy and feature prioritization. All great products deliver a differentiated promise to the market – that defines how your business provides a competitive advantage to the users.

The positioning statement typically communicates the following elements:

***For <target audience>,
<product reference> is a <product category>,
Unlike competitors, we <reasons to buy from you>.***

How to write a position statement:

1. Understand the competition: Analyze competitive brands to gather ideas and brainstorm your unique positioning.
2. Demonstrate empathy: Put yourself in your customers' shoes. Your customer cohort may be different from your competitor's.
3. Reach out to your customers: Send customer surveys, conduct interviews, and examine support data to validate your assumptions.
4. Formulate a concise message: Deliver your message clearly and briefly. Your position statement should convey your brand values. Be authentic, and don't make promises you can't keep.

Here are some examples of well-crafted positioning statements from tech product companies:

1. Spotify: "For music lovers who value a personalized listening experience, Spotify is a streaming service that provides access to millions of songs, curated playlists, and exclusive content, unlike competitors who offer limited or generic recommendations."

Why is this a good positioning statement? Spotify highlights its personalized listening experience, curated playlists, and exclusive content, differentiating itself from competitors who may not provide such tailored recommendations.

2. Zoom: “For businesses and individuals looking for seamless communication, Zoom is a video conferencing platform that offers high-quality video, audio, and screen sharing features, unlike competitors who struggle with connectivity and user experience.”

Why is this a good positioning statement? Zoom emphasizes its high-quality video, audio, and screen sharing capabilities, setting it apart from competitors that may have connectivity and user experience issues.


3. Slack: “Slack is the collaboration hub that brings the right people, information, and tools together to get work done. From Fortune 100 companies to corner markets, millions of people around the world use Slack to connect their teams, unify their systems, and drive their business forward.”

Why is this a good positioning statement? Slack demonstrates its ability and flexibility to address a broad target market.

4. Mailchimp: “Mailchimp is an all-in-one Marketing Platform for small businesses. We empower millions of customers around the world to start and grow their businesses with our smart marketing technology, award-winning support, and inspiring content.”

Why is this a good positioning statement? All-in-one marketing platform Mailchimp positions itself as a friendly, approachable one-stop shop for small businesses.

To spread the word about your competitive positioning, you need an engaging, compelling story. Now let's explore how to craft this story.



Crafting a compelling, clear, and concise positioning statement is a strategic compass, guiding every product decision and feature prioritization, ensuring your product resonates with and fulfills the unique needs of your target audience.

Craft a customer centric story

“Marketing is no longer about the stuff that you make, but about the stories that you tell.”

- Seth Godin

In a world full of noise, people are no longer interested in crafted sales pitches but are receptive to authentic stories. So, what's your product story? Clarifying your message, connecting with customers, and placing them at the center of your brand is the goal of storytelling.

Throughout history, storytelling has proven to be a powerful tool in human communication. An authentic and compelling story is probably the most critical aspect. It triggers a cognitive response and helps you connect with your audience. Your product story should grab attention, elicit emotions, and engage people across different channels.

Fundamentals of your product story:

- 1. Make it a story:** At its core, every story is the same. There is a hero (user), a villain (problem), and a guide (your product!). The hero eventually wins with the help of the guide. If you want your story to stick, try following this convention.
- 2. Make it meaningful:** Your story needs to be exciting and relevant to the people you're trying to reach. It should be something that the audience perceives as high value.
- 3. Make it personal:** Draw your audience into your story. Show how your product improves their life and explain why it's essential to buy your product.
- 4. Make it emotional:** Your story should trigger empathy and hook your audience from the beginning.
- 5. Make it authentic:** Be open, honest, and let your personality shine through. Be consistent to ensure your audience trusts your content.

An example of a good product brand story is Asana, an innovative project and task management platform. Asana's brand story revolves around the hero (team members), the villain (disorganized work and missed deadlines), and the guide (Asana's project management platform).

The story highlights how Asana empowers teams to collaborate effectively, stay organized, and meet their goals. By using Asana, the hero (team members) overcome the villain (disorganization) and achieve success with the help of the guide (Asana).

In another example, HubSpot's brand story focuses on the hero (businesses seeking growth), the villain (ineffective and disjointed marketing efforts), and the guide (HubSpot's all-in-one marketing, sales, and customer service platform). The story emphasizes how HubSpot helps businesses attract, engage, and delight customers, ultimately leading to growth. By leveraging HubSpot, the hero (businesses) conquers the villain (ineffective marketing) with the assistance of the guide (HubSpot's platform).

How to tell your product story?

Now that you know the elements that make a great story, you can create your own product story. Here's a three-step guide to help you:

Step 1: Understand the background of your story Before you start crafting your story, revisit your business goals and understand your customers' needs. Make sure you address these needs in your story to make it more relatable and impactful.

Step 2: Craft the story Great brands put the customer at the center of the story, not the product. Remember, the structure of every story has a hero (customer), the villain (problem), and guide (your product). Focus on the role of your product as the guide that helps customers overcome their problems. Engage with your target audience to position your product as the most appealing guide among your competitors.

Step 3: Share your story The way you share your story affects the content you create. Before you start creating content, identify the best format for your story. Every communication about your product should convey your core brand message. Use various mediums like articles, videos,

infographics, and case studies to share your story. Make sure the content you create reflects your brand personality, voice, and consistent across all channels. Also, ensure the visuals are cohesive with the color palette in your brand guidelines.

An authentic and captivating brand story helps your audience remember who you are, what you stand for, and encourages them to be a part of your story (i.e., purchase your product).



Crafting an authentic, customer-centric story is paramount; it should resonate emotionally, reflecting your brand's essence, making your product memorable and enabling customers to see themselves as the heroes in your narrative.

Identify your minimum viable product

"Build-Measure-Learn feedback loop is at the core of the Lean Startup methodology."

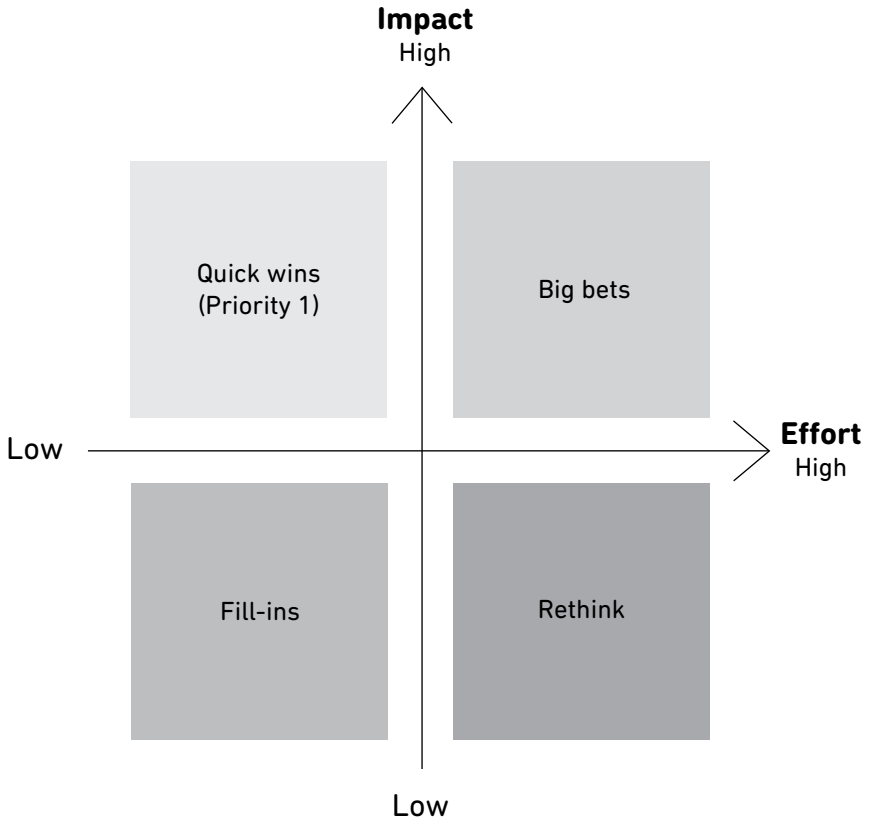
- Eric Ries

In the book 'Lean Startup,' Eric Ries introduces the concept of a Minimum Viable Product (MVP). The MVP is the set of features just enough to deliver value to end users while allowing you to learn and understand the success of the solution. An early MVP test doesn't even need to have real features but can involve the simplest interactions to learn as much as possible. The fastest way to learn is by building successive iterative MVPs.

This approach results in several benefits. Maximizing the learning implies that you put your time and effort into features that customers care about. This reduces wastage and lets you focus on things that are likely to bring revenue. With that said, the MVP shouldn't be an incomplete, unsatisfactory version of the product. Instead, there must be a good balance between the features provided and the value derived.

One crucial aspect of identifying an MVP is prioritizing the features to include in the next iteration. Here's a simple process to help you prioritize those features effectively:

- 1. List the potential features:** Start by creating a comprehensive list of potential features for your product.
- 2. Evaluate user value and team effort:** For each feature, assess the value it brings to the user and the effort required by your team to implement it. User value can be measured by the potential impact on user satisfaction, engagement, or revenue generation. Effort can be estimated by considering the development time, resources, and complexity involved in implementing the feature.
- 3. Rank features based on value and effort:** Once you have assessed the user value and effort for each feature, rank them using a simple scoring system. A common method is the Value vs. Effort matrix. Features that provide high value with low effort should be prioritized, while those with low value and high effort should be deprioritized.



- 4. Create a roadmap:** With your prioritized list of features, create a roadmap for future iterations of your MVP. This roadmap should outline the sequence in which you plan to implement the features, ensuring that those with the highest value and lowest effort are tackled first.
- 5. Iterate and reevaluate:** As you implement features and gather feedback from users, reevaluate your priorities and update your roadmap accordingly.

By prioritizing features based on user value and team effort, you can ensure that your MVP continually evolves and improves, keeping your customers engaged and satisfied while maximizing the return on your efforts. The developed MVP should be rolled out to the target audience and leverage on usage data and feedback to improve the next MVP forming a continuous improvement cycle.

If you want real-world examples of a successful MVP, look no further than the story of Foursquare. Foursquare started with just one feature, which is location-based check-ins. With the validation of their idea on a growing user base, the company successfully launched gamification features, recommendations, and city guides. They chronologically added these features creating a continuous improvement cycle process.

What's the difference between a prototype and an MVP? The objective of a prototype is to collect user feedback while the purpose of an MVP is to understand user behavior through real user interactions. While a prototype is just a simulation to a selected set of users, an MVP is typically a market rollout.

After gathering valuable insights from your MVP, the next critical step is to make the go/no-go decision. This decision will determine whether you should proceed with further development, pivot your product strategy, or, in some cases, abandon the project altogether. To make an informed decision, evaluate the data and feedback collected from the MVP process and consider the alignment with your overall business objectives and resources.

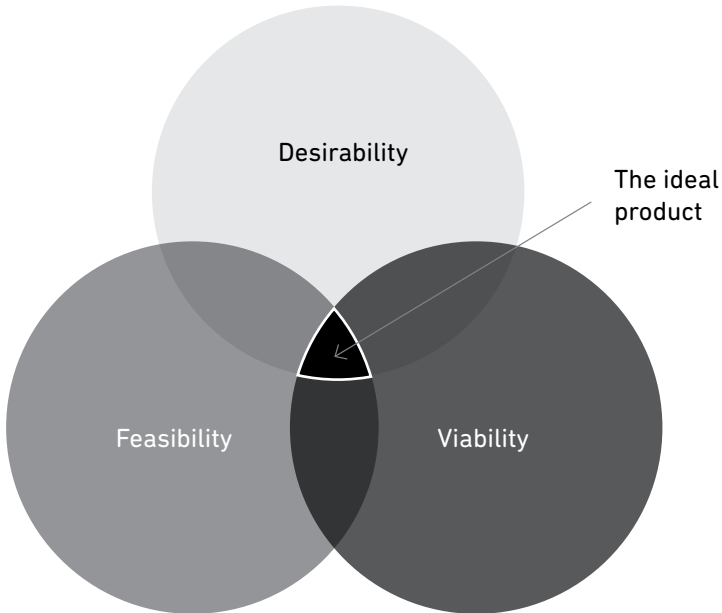
Identifying your Minimum Viable Product is about discerning and delivering core value swiftly and efficiently, allowing learning and refinement through real-world interactions, ultimately paving the path for product success.

Make the Go/No-Go decision

“What is now proved was once only imagined.”

–William Blake

Now that you have looked at the problem and solution in detail, it is time to revisit and summarize your learning to make a Go/No-go decision. This checkpoint is important as your investments beyond this phase will be significant. The right decision lies at the intersection of desirability, viability, and feasibility.



You must check if the product you envision meets the following criteria:

Desirability: This criterion focuses on the demand and appeal of your product to potential customers. Here are some factors to consider:

- Is the problem real? Validate if the problem you're addressing exists and if it's significant enough for users to seek a solution.
- What brings customers through the door? Identify the key features or unique selling points that attract users to your product.
- What keeps them coming back? Understand the aspects of your product that create customer loyalty and encourage repeat usage.
- Will they buy it from you? Analyze the market landscape and determine if your product offers a compelling enough value proposition for customers to choose it over competitors.

Viability: This criterion pertains to the financial aspects and potential profitability of your product. Consider the following factors:

- Is the idea profitable? Examine the cost structure, pricing, and revenue streams to determine if the product can generate a profit in the long run.
- Where do customers want to spend most? Identify the key aspects of your product or service that customers value the most and are willing to pay for.
- Will the project give you the return on investment (ROI)? Calculate the expected ROI by comparing the costs of developing and launching the product against the potential revenue it can generate over time.

Feasibility: This criterion assesses the practicality of successfully developing and launching your product. Keep these factors in mind:


- Are people, tools, technology, and other resources available? Evaluate if you have the necessary team, tools, and technology in place to build and support the product.
- Can you deliver on a scale when the idea catches on? Assess your capacity to scale the product to meet growing demand without compromising quality or user experience.
- Are there any technical, legal, or regulatory constraints that may hinder the development or adoption of your product? Research potential barriers and plan for how to overcome them.

By thoroughly evaluating your product against these criteria, you can make a more informed decision on whether to proceed with its development or pivot to another idea.

Several tech products have been abandoned after the MVP stage due to the lack of desirability, viability, or feasibility. One example is Google Wave, a communication and collaboration tool that aimed to merge email, instant messaging, and social networking. Despite its innovative features, it failed to resonate with users due to its complex interface and lack of a clear value proposition. As a result, Google decided to discontinue the product in 2010.

Another example is Amazon's Fire Phone, launched in 2014, which aimed to compete with the likes of Apple and Samsung. However, the phone's unique features, such as dynamic perspective and Firefly, did not provide sufficient value to consumers, and the phone struggled to find its market. Amazon eventually discontinued the Fire Phone just a year after its launch.

If it's a 'Go', the journey gets even more exciting in the days ahead. Read on to see how you can plan your execution. Have fun!



A Go/No-Go decision is pivotal; it harnesses insights from desirability, viability, and feasibility assessments to determine whether the envisioned product holds the promise of success or necessitates a strategic pivot.

Phase 4

Plan your journey

| | |
|------------------------------------|----|
| Derive ballpark estimates | 72 |
| Develop a product roadmap | 76 |
| Evolve high-fidelity prototypes | 79 |
| Establish a pricing strategy | 82 |
| Create a cohesive brand identity | 86 |
| Validate your product-market fit | 89 |
| Automate your delivery pipeline | 92 |
| Invest on a quality-driven culture | 95 |

Having gained a deep understanding of the solution during the Immerse phase, you'll now begin to develop initial plans for its implementation. Keep in mind that these plans are living documents and should be updated throughout the product's lifecycle. The key is not to create a rigid, detailed plan upfront, but to establish an agile planning mechanism.

During this phase, you'll work on creating an initial product roadmap, developing marketing collateral, determining a pricing strategy, and detailing your approach to the next phase - Build. By maintaining flexibility and adaptability, you'll be better prepared to navigate the market dynamics that arise throughout your product's journey.

Derive ballpark estimates

"Estimation is an invaluable tool for anticipating and managing the project realities of time and cost."

- Joseph Phillips

Estimating the effort and resources required for a digital product is crucial for planning, budgeting, and setting realistic expectations for stakeholders. You may use several estimation techniques such as expert judgment or analogy-based estimation.

Expert Judgment: Expert Judgment is an estimation technique that relies on the knowledge, experience, and intuition of subject matter experts. These experts have a deep understanding of the domain and the tasks involved in the project, which allows them to provide educated estimates. This technique is highly dependent on the expertise of the individuals involved and can be prone to biases, such as overconfidence or anchoring. However, it is often used when there is limited data available, or when the project is unique and challenging to quantify. To minimize biases and improve the accuracy of expert judgment, it's crucial to involve multiple experts and encourage open discussion and consensus-building.

Analogy-based Estimation: Analogy-based Estimation, also known as Comparative Estimation, is a technique that involves comparing the current project with similar, completed projects to derive estimates. This approach assumes that projects with similar characteristics will require similar efforts and resources. The key to successful analogy-based estimation is identifying relevant and accurate historical data and adjusting for any differences between the past projects and the current one. This technique can be more accurate than expert judgment, especially when there is a wealth of historical data available. However, it may not be suitable for highly innovative or unique projects that have no comparable precedents.

In this chapter, we will discuss a technique called Story Point Estimations which is a specialized technique used primarily within Agile Development Processes. Story Point Estimation can be viewed as a hybrid approach that combines elements of both Expert Judgment and Analogy-based estimation techniques. In

this method, team members draw upon their expertise, experience, and intuition to assess the complexity and effort involved in completing user stories relative to a baseline, like the process of gathering input from multiple experts in Expert Judgment. Additionally, they make comparisons to previously completed tasks or projects, as done in Analogy-based estimation, to better understand the relative complexity and effort involved.

You should start the Storypoint estimation process by breaking down the scope into manageable units such as epics, features, and user stories.

Epics: An epic is a high-level, broad objective or goal that represents a significant piece of work in the product's development. Epics are typically large and require further breakdown into smaller units (features and user stories) to be actionable.

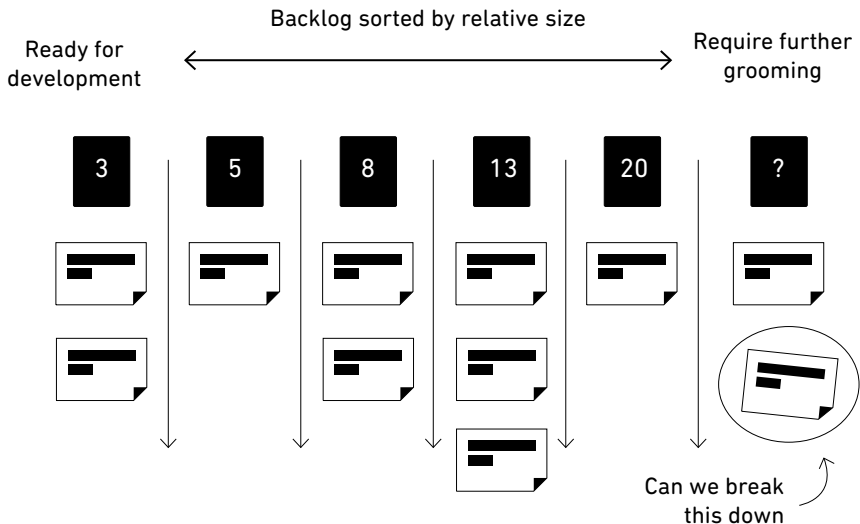
Features: A feature is a functional component of the product that provides specific value to the user. Features are derived from epics and can be broken down further into user stories.

User Stories: A user story is a concise description of a specific requirement or goal, written from the perspective of the end-user. User stories help bridge the gap between product, engineering, and development teams by fostering collaboration and ensuring everyone is on the same page. With user stories defined, the next step is to estimate the effort and resources required to complete each story. With Story point estimation approach, you focus on comparing the size and complexity of user stories rather than attempting to determine the exact duration or resources required.

The process of story point estimation typically involves the following steps:

1. **Use a predefined scale:** Teams often use a predefined scale to assign story points, such as the Fibonacci sequence (1, 2, 3, 5, 8, 13, 20, etc.) or the T-shirt sizing method (XS, S, M, L, XL, etc.). Using a predefined scale helps to maintain consistency and avoid assigning arbitrary values.

- 2. Create a baseline:** The team selects a user story that represents a simple, medium sized, yet meaningful piece of work, which will serve as the baseline or reference point. This user story is then assigned a story point value, often a medium number.
- 3. Compare other stories:** The team then compares other user stories to the baseline, evaluating their complexity and effort relative to the reference story. The team assigns a higher story point value for more complex stories and a lower value for simpler ones.
- 4. Reach a consensus:** The team members discuss their individual story point estimates for each user story and reach a consensus. Techniques such as Planning Poker or the Wideband Delphi technique can be used to facilitate this.



Once you have determined the story point estimations for all user stories, you can derive the development cost. Now, you'll need to determine the team's velocity, which is the number of story points the team can complete in a given period, such as a sprint or a month. By dividing the total story points by the team's velocity, you can estimate the number of iterations required to complete the project. To arrive at the development cost, you'll have to consider the cost per

iteration, which includes salaries, overheads, and any other expenses related to the team. Multiplying the cost per iteration by the total number of iterations will give you the estimated development cost for the project.


To derive at the total cost of the product development apart from the development cost, you should also factor other costs associated with the software product such as:

Hosting: Consider the cost of hosting your digital product on-prem or on cloud platforms like AWS, Azure, or Google Cloud. This includes the costs for storage, compute, and data transfer.

Tooling and Licenses: Account for the cost of software tools, licenses, and subscriptions required for the development, testing, deployment, and monitoring of the digital product. This could include project management tools, version control systems, and testing tools.

Operational Costs: Factor in the costs of physical infrastructure, communication lines, support and maintenance, including ongoing bug fixes, security updates, and infrastructure management. This may also involve costs related to customer support, staff training, and product documentation.

As the project progresses, it's crucial to revisit and refine your estimates based on actual experiences, lessons learned, and changing requirements. This approach will help improve the accuracy of your estimates and allow for better decision-making and resource allocation.



Deriving accurate estimates involves employing structured techniques and accumulating insights, serving as the backbone for resource allocation, strategic planning, and setting the stage for the successful realization of digital products.

Develop a product roadmap

"The true mark of a leader is the willingness to stick with a bold course of action - an unconventional business strategy, a unique product-development roadmap, a controversial marketing campaign - even as the rest of the world wonders why you're not marching in step with the status quo."

- Bill Taylor

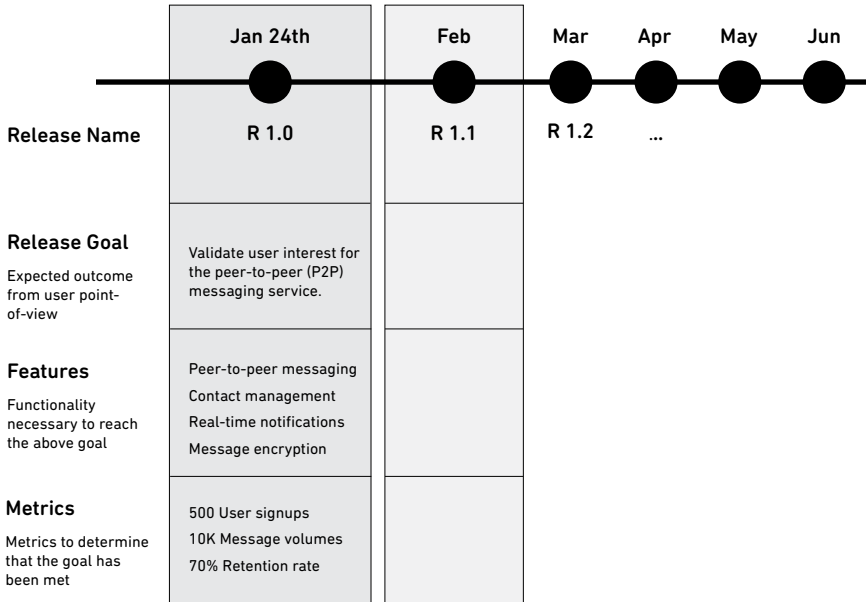
Having received the 'Go' signal, it's time to connect the product's vision with a timeline that the product team can bring to life. This is known as the product roadmap, a visual way to quickly communicate a plan or strategy. It serves as a bridge between the product strategy and execution, enabling collaboration between engineering and product management teams.

A well-crafted product roadmap should communicate the following strategic elements:

1. **Strategy & Vision:** Explain how the initiatives align with higher-level product goals and the overall product and business vision.
2. **Resources:** Detail how the team will achieve these goals and the resources required.
3. **Time Estimates:** Provide a timeline for important deliverables.
4. **Dependencies:** Identify the teams and team members that need to be involved and why.

To build the product roadmap, list and prioritize key milestones in the product journey. Next, use the ballpark estimates to determine the timeline. Consider effort estimates and resource availability when determining the timeline. The product roadmap should reflect the timeline for both user features (market demands) and platform features (engineering needs). Technical concerns, such as DevOps and test automation, should not be ignored in the process. Despite market pressure, prioritize these technical concerns, as they can be costly to fix later.

Involving multiple contributors from your engineering and leadership teams in the milestone creation process ensures greater buy-in to the plan. Keep in mind



that these milestones will likely be refined as development progresses and actual experiences are gathered.

Creating Stakeholder Alignment

It's important to remember that a roadmap is a communication tool, and different audiences may require different information. Therefore, multiple versions of the roadmap may be necessary. For example, internal stakeholders might need more detailed information, while external customers should be presented with high-level features and estimated timeframes.

The product roadmap is an excellent tool for customer communication. It creates anticipation for upcoming features and releases, allowing customers to align their business activities with your release milestones. By consistently updating and communicating the roadmap, you can prioritize features based on customer needs and expectations.

In summary, a product roadmap is an essential tool for planning, strategizing, and communicating a product's direction. Remember, even the best estimates remain estimates. Be mindful when treating these as commitments and keep flexibility to adapt as needed.



A product roadmap is a visionary alignment tool, bridging strategy and execution, ensuring collective insight and flexible adaptability in narrating a product's journey to realization.

Evolve high-fidelity prototypes

"Prototyping is the conversation you have with your ideas."

- Tom Wujec

No matter how great your intended product features are, finding usability issues or design flaws after a functional build would be expensive. So, let's have a look at what can be proactively done to reduce this risk of disappointing your users.

You've already validated your ideas through low-fidelity prototypes as discussed previously. Now, it's time to turn these wireframes into interactive, high-fidelity prototypes, i.e., a clickable prototype closely resembling your final product.

For example, when introducing new features, Evernote often creates high-fidelity prototypes that allow them to test various note-taking features, explore user interactions, and optimize the organization and retrieval of information.



Image Credit: <https://medium.com/@svalentgoed/add-a-feature-ironhack-637e9db4a8e7>

Look at your low-fidelity wireframes and identify all the interactions and transitions you want to create between those screens. All the user flows and user journeys you have put together under the Explore phase would help you do this. The objective of this exercise is to know the best possible workflow to make it a delightful experience.


The next step is to find a suitable software prototyping tool. There are many paid and free tools out there, such as Sketch, XD, Figma, and InVision. You may want to evaluate their features, pricing, ease of learning, system requirements, provided collaboration capabilities, and other features before selecting the right tool for you.

Once developed, this clickable prototype can be given to some of your potential users to obtain their feedback on design, ease-of-use, functionality, and user experience. You can also use this to gain accurate insights into how a user would engage with your product and further refine the prototypes. This iterative refinement can be done through multiple rounds until you achieve the optimal user experience. In addition, you can use these prototypes to perform user experience experimentation with techniques like A/B Testing.

Further, high-fidelity prototypes can be effectively used to communicate the design and functionality of your product to other stakeholders, such as potential investors, sales and marketing staff, developers, etc.

For example, when Buffer, the social media management tool, was first being developed, the founders created a simple landing page with a prototype to showcase their idea. This allowed them to gauge interest, gather feedback, and secure early adopters even before the product was fully developed.


The primary intention here is to make sure that you get the acceptability of functionality and design before you invest a lot of time in the actual development. Done right, an effective prototype may even allow you to lock in future revenue without writing a single line of code.

Tweet more consistently with  **buffer**

- 1 Choose times to tweet.**
For example, 3 times a day at 9:30, 13:30 and 17:30.
- 2 Add tweets to your buffer.**
Manually or with our handy browser extensions.
- 3 buffer does the rest. Relax.**
We tweet for you. Just keep that buffer topped up!

[Plans and Pricing](#)

© 2010 buffer. All rights reserved.

Tweet more consistently with  **buffer**

Hello! You caught us before we're ready.

We're working hard to put the finishing touches onto buffer. Things are going well and it should be ready to help you with Twitter very soon. If you'd like us to send you a reminder when we're ready, just put your email in below:

[Remind me](#)

© 2010 buffer. All rights reserved.

"Evolving high-fidelity prototypes breathe life into ideas, ensuring the envisioned product is user-aligned, functional, and refined, allowing foresight into user interaction before the tangible creation begins."

Establish a pricing strategy

"Price is what you pay. Value is what you get."

- Warren Buffett

Now that you have an idea of the digital product you want to take to market, it's time to start thinking about your pricing strategy. When determining the pricing strategy, consider the value your product provides to its users and the market's willingness to pay.

Assigning your product an accurate price is not a straightforward exercise or a one-time activity. Many questions will come to your mind. For example, should I price it low to attract more customers? Should I price it to maximize profits? Should I charge a one-off fee or a subscription payment?

Let's break this down into a few areas to help you derive your price points. First, let's look at the pricing models available out there and the type most suitable for your product.

There are two broad types of software product revenue models:

- Perpetual (one-time payment with maintenance fee) – the customer owns the license indefinitely.
- Subscription (recurring payments) - software delivered as a service, and customers pay for usage monthly or annually.

From the customer's perspective, perpetual licenses are seen as a significant investment upfront, while subscription-based payments are considered operational expenses. In today's context for both customers and product companies, the subscription-based model is preferred due to its benefits, as highlighted below.

For the customer:

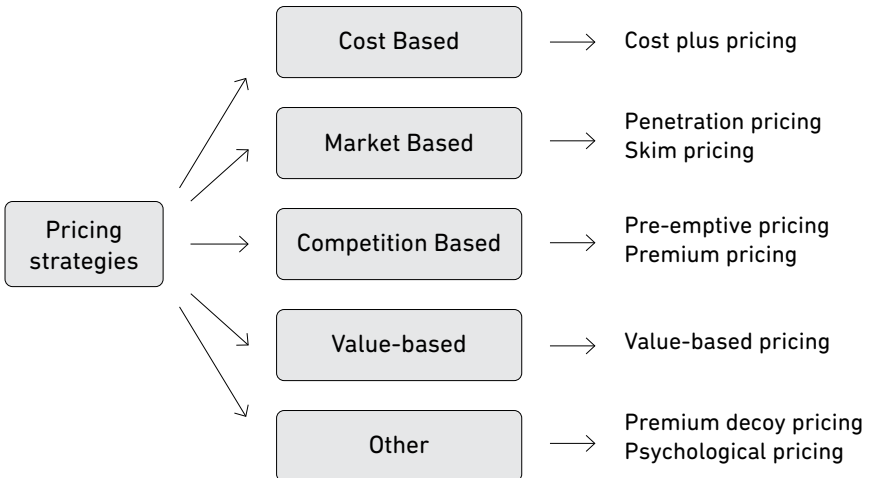
- Lower barrier to get onboard the product as the cost is spread over a period
- Payments can be considered as OPEX not CAPEX
- Simpler terms and all-inclusive fees
- Software updates are inclusive in the subscription

For your product company:

- Investors prefer this due to long-term recurring revenues
- Able to maintain an ongoing relationship with the customer to refine the product
- Able to acquire a long tail of customers due to lower entry barrier
- A notable example is Adobe, which shifted from perpetual licensing to a subscription-based model with its Creative Cloud suite. This change allowed the company to generate more stable revenue and provide regular updates to its customers.

Companies that follow the subscription pricing model often continuously adjust the price to meet business objectives. For example, you can increase the price for higher margins or reduce the price to gain market share depending on the company strategy. Maintaining a positive ratio between lifetime value per customer (LTV) and the customer acquisition cost (CAC) indicates a sustainable pricing strategy. If not, you lose money every time you acquire a new customer.

Here are some of the price positioning techniques that are commonly used, along with some examples.



Cost-plus pricing: In this approach, you include a margin on top of the cost of the unit. A small software development company might use this strategy to ensure a profit on each project while staying competitive in the market.

Penetration pricing: Lower the price to attract customers at the beginning and raise the price once the product is established in the market. For example, when Netflix first entered the streaming market, they offered low subscription prices to gain a large user base before gradually increasing prices as their content library expanded.

Skim pricing: is a pricing strategy that sets new product prices high and subsequently lowers them as competitors enter the market. Skim pricing is the opposite of penetration pricing, which prices newly launched products low to build a big customer base at the outset.

Pre-emptive pricing: Pricing is set way below the market prices to attract customers from your competitors. Amazon Web Services (AWS) used this strategy by offering low-cost cloud services to gain a significant market share early on.


Premium pricing: To reflect the premium value of your product, set the pricing at a higher point. Luxury brands like Tesla and Rolex employ this strategy to signify their products' high quality and status.

Value-based pricing: Find out how much customers are willing to pay for the value of your product before it becomes a deal-breaker and price accordingly.

Psychological pricing: Set the pricing values to irregular values where customers are psychologically willing to buy at (e.g., \$19.99). Retailers like Walmart often use this technique to create the perception of a discount. Premium decoy pricing: Set the price at a higher price point to encourage the purchase of another product. For instance, Apple often introduces higher-priced iPhone models to make the base model seem more affordable in comparison.

Throughout your product journey, you may need to mix and match different pricing strategies. Subscription-based products often lean towards value-based pricing, supported by thorough market research. It's essential to develop a pricing strategy that considers the product's perceived value by the customer. Identify target markets and customer personas and determine what features they desire at each price point. Start with pricing that aligns with the market and the product's perceived value, and don't hesitate to experiment along the way.

Potential customers might struggle to pinpoint specific prices they are willing to pay for a set of features. However, asking them to indicate price ranges for the most expensive, lowest, and best bargain price levels can yield valuable insights. Validate your pricing assumptions by engaging with your prospects, conducting market research to understand competitors' pricing for similar products, and segmenting customers into groups to test their willingness to pay for various features. This approach will help you better understand how to package your product for different market segments.



Strategically tailoring your pricing strategy, through informed evaluations and adaptive methods, can propel your product's market appeal and financial sustainability, aligning value perception with user willingness to pay.

Create a cohesive brand identity

“Your brand is what other people say about you when you're not in the room.”

- Jeff Bezos

You might wonder, “Why do brand guidelines matter?” and “What’s the connection between the success of a digital product and setting up brand guidelines?” The answer lies in the power of consistency. Brand guidelines help you consistently build, differentiate, and communicate your brand. Therefore, it’s essential to consider your product’s visual identity early in the process, including the product name, logo, visual style, and associated colors. For example, when Apple launches new products, they maintain a sleek and minimal design that resonates with their target audience. This consistent design approach has played a significant role in making Apple one of the most recognizable tech brands worldwide.

Creating a brand guideline document ensures that anyone can read and understand the essential elements of your identity and why your product exists. Identifying crucial visual elements such as typography, colors, and visual style early in the development process makes it easier to design the user interface. Doing so also prevents potential rework and ensures a more seamless experience for the entire product team.

A brand is the overall impression and experience you give to your audience and consumers, it's not just a logo. Your brand expresses the value you provide. It's you!

- Amy Locurto (Brand Strategist)

If you’re building a product within an established business, consider whether the new product’s brand should complement the existing portfolio or remain independent. For example, Google’s suite of products, including Gmail, Google Drive, and Google Docs, all share a similar branding style with recognizable colors and logos. This consistency helps create a cohesive brand identity across the entire product line.



Google experimenting with its brand identity

Image Credit: <https://design.google/library/evolving-google-identity/>


| | | | |
|---------------------------------|--------------------------|--|-------------------------------|
| Logos Logo | Colors | Interactive elements | Hierarchy |
| Photography | Tone of voice | Fonts and typography Typography Typography Typography | Data visualization |
| Iconography | Illustration | Video and motion | Web design |

Image: Some elements that should be included in a brand guide

Your brand guidelines should be used both internally and externally to ensure uniformity and continuity of your brand's visual identity and tone of voice. This crucial document should outline your brand strategy, brand story, and brand promise, communicating the essential elements that define your product's identity.

For example, Slack has done an exceptional job in establishing their brand identity through well-defined guidelines. Slack's branding features a playful and colorful design with a distinctive hashtag logo. These well-established brand guidelines publicly published under 'Slack Brand Center' have helped the company create a strong and memorable presence in the competitive SaaS market.

By establishing brand guidelines early in the product development process, your software product will have a consistent look and feel when you launch, avoiding the impression that elements have been hastily added on at the end. This approach will not only enhance your product's overall appeal but also contribute to building a strong brand identity that resonates with your target audience.



Establishing cohesive and compelling brand guidelines from the outset fosters a unified product identity, enhancing recognition and resonance, and steering the visual and communicative essence of your brand consistently.

Validate your product-market fit

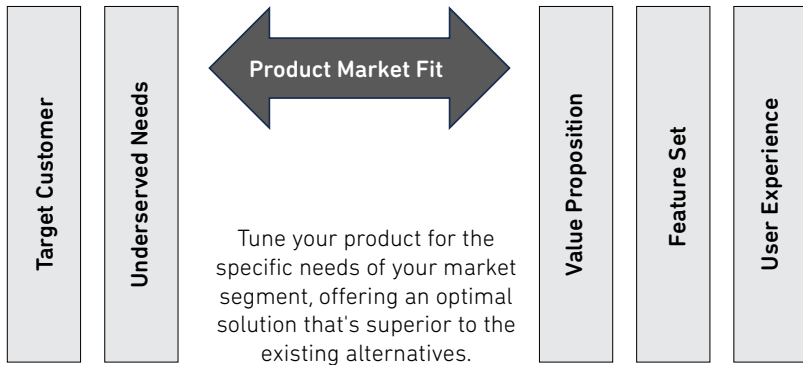
“Product-market fit means being in a good market with a product that can satisfy that market.”

- Marc Andreessen

Now that you've defined your brand position, it's essential to determine how appealing your offering is to the target market. This is done through optimizing your product-market fit.

Product-Market Fit vs. Problem-Solution Fit

It's crucial to understand that the product-market fit is different from the problem-solution fit. When you effectively identify a solution to a real problem, you have found the problem-solution fit. However, that doesn't mean your solution is appealing to everyone who has the problem.



You may need to fine-tune your solution to fit the unique needs of your target market segment. It's time to see if small adjustments can make your product attractive to your target audience.

Here are some examples of good product-market fit from digital product companies.

Wave: Wave was a hugely successful accounting platform aimed at small businesses and freelancers in the US and Canada. As their customer base grew across the globe, they made a strategic decision to narrow their product-market fit further to serve only their target market in the US and Canada. This allowed them to focus on more specialized features important to their target customer segment, such as localized tax compliance and payment processing, thereby increasing the product's value.

Canva: Canva, the online graphic design platform, found its product-market fit by addressing the challenges faced by non-designers, marketers, and small business owners who needed to create visually appealing content. By offering an intuitive drag-and-drop interface, a vast library of templates, and a wide variety of design elements, Canva empowered users to create professional-looking designs without any prior experience. The platform's focus on simplifying the design process and making it accessible to nonprofessional users helped Canva establish a strong product-market fit and attract millions of users worldwide.


Notion: Notion, the all-in-one workspace app, identified a need for a unified solution that could replace multiple productivity tools, targeting startups, small businesses, and remote teams. By offering a highly customizable platform that seamlessly combines note-taking apps, task managers, and databases, Notion managed to create a product-market fit which appeals to tech and engineering teams. The platform's emphasis on flexibility and adaptability has contributed to its rapid growth and adoption across various industries.

How to measure product-market fit

There's no well-defined method to measure product-market fit, but there are ways to assess whether you're on track. However, the Pirate Metrics model (AARRR) is a useful framework to assess this.

- Acquisition: How quickly do customers make up their minds about a purchase?
- Activation: Once registered, do they engage in using the product?
- Retention: What is your customer retention rate or churn rates?
- Referral: Do your users actively promote your product?
- Revenue: How much revenue and profit do you generate?

Analyze the questions above across your customers and lead funnel to identify the target segment with the highest stickiness for your offering. Focusing on finetuning your product-market fit to exploit this segment will give you the best returns. A good product-market fit will let you set yourself apart from your competition.



Finding your product-market fit means not only solving a problem but also refining your solution to align perfectly with the unique needs of your target audience, ensuring your product resonates and differentiates itself.

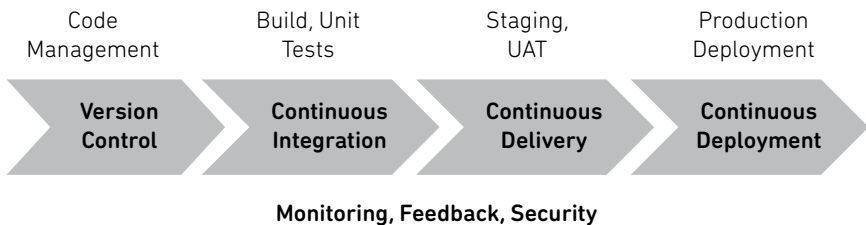
Automate your delivery pipeline

“Continuous Delivery is a software development discipline where you build software in such a way that the software can be released to production at any time.”

- Martine Fowler

Imagine working hard to release a new version of your product. You have spent time and effort to get it to the end users, but due to a human error, you end up with an expensive rollback. The risk of prolonged downtime and revenue loss can be reduced by introducing Continuous Integration, Continuous Delivery, and Continuous Deployment. The steps required for the code to make its way to production can be called a delivery pipeline. These automated pipelines enable you to deliver frequently and predictably, thereby reducing human errors.

What are the steps in the pipeline? To identify the steps, we must first identify where the delivery pipeline starts and where it ends. The pipeline begins as soon as the developer pushes the code for a feature to the central repository. Along the pipeline, first, the code is integrated, next it is tested, then the required infrastructure is provisioned, associated data migrated, and the functionality verified before deployment to production. It is completed once the feature is released to all the end-users.



Continuous Integration

Continuous integration is the process of continuously merging code changes into a central repository. This ensures that there is no scatter of code, and the central branch is up to date with conflict free code. You must implement a good branching strategy to enable parallel programming yet minimizing merge conflicts.

Consider automating tests with quality gates to maximize the code quality and process efficiency. These gates could range from automated static code analysis, developer unit tests, end-to-end UI tests and manual peer code reviews. At each quality gate, you may have automated and manual approvals to proceed to the next level.

Continuous delivery

Continuous delivery is an extension of continuous integration. It focuses on automatically building code and deploying the build artifacts to environments such as QA and staging. You must consider using Infrastructure as Code (IaC) to provision the underlying infrastructure as part of this process.

To practice this effectively, automate repetitive tasks to free up developers for more value-adding activities. In addition, work in short release cycles to deliver frequent updates to test environments, allowing faster feedback cycles to address development issues.

Continuous deployment

In a competitive market, it is crucial to ensure the availability of a continuous stream of enhancements and new features. Continuous deployment is the process of automatically deploying the build artifacts from the testing phase to production. This not only improves productivity and minimizes errors but is also less risky as you deploy smaller chunks of changes at a time. Since the changes are minimal in each release, even if an error occurs, it can easily be rolled back to the previous state.

Etsy, an e-commerce platform, is an excellent example of continuous deployment. They deploy their codebase multiple times a day, ensuring that new features and fixes are available to users as soon as they are ready. When deploying new releases to production, it is crucial to minimize interruptions and downtime for users. In the past, users were willing to tolerate a couple of hours of downtime, but modern users expect continuous application availability even during a new release. To meet these expectations, deployment strategies such as blue-green or canary can be employed.


Blue-green deployment

Blue-green deployment involves maintaining a clone of the production environment, called “green,” which remains idle. When a new feature is ready for release, it is deployed to the green environment and tested for functionality and performance. If all tests are successful, users are switched from the “blue” (running) environment to the green one. This strategy reduces downtime and enables a quick rollback if any problems are found in the new release.

Canary deployment

Canary deployment, on the other hand, is less risky than blue-green as it switches only a percentage of users to the new features, rather than changing all users at once. This approach is used to test high-risk features, as it only affects a selected group of users. For example, Google releases new features to different user groups in stages, starting with the development team, then within the region, free users, and finally paid users.

If your product has a global reach, deployment becomes even more complex. You will need to consider factors such as time zones, localization requirements, and the availability of resources and support staff needed for a successful deployment. By implementing modern deployment strategies, you can ensure a smooth transition for your users and maintain high levels of application availability during updates.



Automating your delivery pipeline through Continuous Integration, Continuous Delivery, and Continuous Deployment not only reduces errors but also allows for quicker and more frequent releases, keeping your digital product competitive and responsive.

Invest on a quality-driven culture

“Quality is never an accident; it is always the result of intelligent effort.”

– John Ruskin

Some assume Quality Assurance (QA) as a narrow slice of work—something that happens between writing code and deploying it to production. But software quality doesn't happen at just one point in time, and it encompasses more than your testing tools or bug count. Quality is a mindset, it's key to building a quality product, and it's a culture that your team — and the entire company — should be involved in.

A culture of “quality” is based on the premise that no single tool, test, or person can guarantee quality. To be a substantial part of the company culture, everyone needs to support and contribute—from management to architects and developers.

Here are a few tips to get closer to establishing a quality culture.

Provide actionable data: Having solid metrics (the impacts of bugs, what matters to users?) tied to actual end-user satisfaction indicators will bring lots of credibility to quality concerns. Amazon tracks various metrics, such as page load time and error rates, to understand the impacts of bugs and the importance of various features for users. This actionable data helps the company prioritize bug fixes and feature development based on customer needs, ensuring a high-quality user experience.

Align stakeholders: Providing actionable data to the right influencers will inform decision-makers that drive quality across the organization. To do this, Quality Engineers need to be at the table for cross-functional meetings and processes. Atlassian, the company behind tools like Jira and Confluence, ensures that Quality Engineers are part of cross-functional teams, enabling them to participate in decision-making processes and effectively communicate quality concerns with stakeholders, such as product managers and developers.

Have clear policies that advocate for quality: Define best practices and policies that promote quality deliveries. For example:

- Define ready: Ensure this includes criteria that specify whether requirements are testable and clearly state the users' expected outcome (acceptance criteria) before the team attempts to provide a solution.
- Define done: Describe criteria whether all validation activities have been completed with acceptable results.

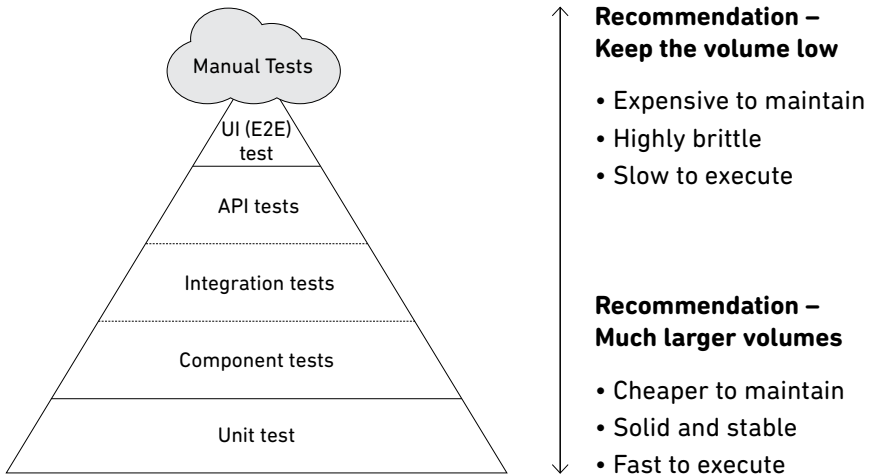
Invest in the right tools and infrastructure: Invest in the tooling and infrastructure that will deliver repeatable and reliable results. Microsoft invests heavily in the right tools and infrastructure for quality assurance. They use a combination of in-house and third-party tools to create a robust testing environment. This investment enables Microsoft to consistently deliver high-quality software products by automating repetitive tasks, reducing human error, and facilitating collaboration among team members.

An effective QA and testing strategy should align with a product's quality objectives by addressing feature correctness, defect analysis and prevention, and risk management. This involves evaluating if features meet requirements and non-functional aspects, identifying root causes of defects to prevent future issues, and assessing risks to plan appropriate test strategies. By focusing on these key aspects, teams can ensure a robust and comprehensive approach to maintaining product quality.

Test automation

Test automation plays a crucial role in efficiently addressing repetitive and time-consuming tasks during the development of digital products. By automating these tasks, teams can save time, reduce human error, and concentrate on more valuable activities, ultimately leading to a more streamlined and effective QA and testing process.

Test automation also provides instant feedback on the integrity of software products as new features are added, shortening the feedback loop and aligning with agile practices, Continuous Delivery, and DevOps culture. The "Test Pyramid" metaphor helps categorize software tests into different levels, emphasizing the importance of having more unit tests and fewer UI tests (e2e tests). User interfaces (GUI) are fragile and susceptible to change,



making UI tests costly and challenging to maintain. In contrast, unit tests are closer to developers, more cost-effective, and faster to execute.

As you move up the testing pyramid, the total cost of ownership increases. Deciding the appropriate level of automation involves adopting a risk-based strategy, pushing tests down the pyramid rather than writing them all at the top. A common and effective approach involves maintaining a ratio of 70% - unit tests, 20% - integration tests, and 10% - UI tests.

Maximizing test automation allows QA engineers to focus on more valuable activities and gives developers confidence when making code changes, knowing they have a safety net. The implementation of test automation will be discussed in a later section.

A culture of quality is not merely a step in the development process; it's a mindset embraced by the entire team, focused on providing actionable data, aligning stakeholders, establishing clear quality policies, and investing in the right tools and infrastructure.

Phase 5

Build and validate

| | |
|---------------------------------|-----|
| Prioritize the product features | 102 |
| Establish a design system | 106 |
| Take control of code quality | 108 |
| Embrace test automation | 111 |
| Shift-left your quality gates | 114 |
| Establish a SecOps practice | 117 |
| Protect personal data | 119 |
| Setup your support processes | 123 |
| Delight customers at onboarding | 126 |

Now, it's time to bring your plan to life. As you advance beyond this phase, the cost of change will rise due to the involvement of various stakeholders, such as users, and the resulting technical dependencies.

To avoid costly mistakes, treat your requirements as assumptions that need validation in your target market. Use this phase to experiment, gather feedback, and ensure business alignment across all assumptions. Validation can take various forms, including Minimum Viable Products (MVPs), demos, user testing, and architectural experiments. Consider engaging pilot customers as testers to verify your primary value propositions.

During this phase, you'll prioritize and develop functionalities, establish quality procedures, and focus on customer onboarding. By carefully managing each aspect of the Build phase, you'll set a strong foundation for your product's success.

Prioritize the product features

"Innovation is saying 'no' to a thousand things."

- Steve Jobs

Effectively prioritizing product features is crucial as you always operate within limited budgets and time constraints. Inadequate prioritization can lead to missed or delayed essential features, resulting in lower customer satisfaction and wasted effort.

Transitioning from an opinion-driven to a data-driven prioritization process requires identifying various sources that contribute to feature requests. These sources can include user feedback, input from sales and marketing teams, suggestions from the engineering team, competitor analysis, product vision alignment, investor interests, and insights from domain experts. By understanding these diverse sources and establishing a process to obtain and consolidate information, you can make more informed decisions when prioritizing product features.

Beware of the pitfalls of prioritization

There will always be many opinions about what should be built next. If you are pulled in many directions, it can be disorienting to the team. This is where a proper process can help you to make clear decisions on prioritizations. Answer the questions below to avoid falling into common pitfalls of prioritization.

- Are you prioritizing without considering business goals and prioritized personas?
- Are you prioritizing features without considering the product roadmap?
- Do you prioritize based on someone's immediate gut reaction?
- Are you prioritizing the easiest first without considering value or vice versa?
- Is your technical backlog getting neglected in the rush to develop features?

- Are you only listening to the loudest stakeholder or the biggest customer?
- Are you only chasing after your competition and becoming a follower?

If your answer is YES to any of these questions, your approach to prioritization may be ineffective. The effectiveness of a prioritization method depends on the current phase of your product. Remember, it's not appropriate to apply the same technique at all phases of development.

Prioritizing at early phases

At the early discovery phase, you may not have ample data or resources to rely on, and most of your ideas are still hypotheses. The key to effective prioritization is to focus on specific business goals of your product, which will help drive your attention to the target users who share those goals. By identifying the goals and activities that your users want to undertake, you can then determine the most relevant features to develop.

To validate priorities and gain insights, seek out signals from your target market or conduct desk research without excessive effort. Send out surveys, conduct interviews, and listen to your users to understand their needs and goals. This approach helps avoid the common prioritization error of focusing on features first, instead of emphasizing the product's business goals and the users who have those goals. Your plan should aim to deliver the 'highest business value' with the 'lowest effort' by aligning feature development with user goals and activities.

To maintain focus on the highest-priority features, it's crucial to say "no" to some ideas. While the MoSCoW method (Must-have, Should-have, Could-have, and Won't-have) is a more traditional, waterfall-style approach, there are alternative techniques that can be employed to prioritize features.

One such method is dotmocracy, which involves your potential users and team members voting on their preferred features using dot stickers, thus generating a visual representation of feature importance. Alternatively, you can tag stories as "Must Haves", "Delighters", or "Satisfiers" to categorize features based on their impact on user satisfaction and overall product

success. These more flexible methods facilitate a prioritization process that's better suited to agile development environments.

Prioritizing at MVP phases

Involving customers in the prioritization process is crucial, especially once you have developed an early version of your product. Focus on selecting features that provide the most significant learning opportunities from your users. To gather valuable feedback for future prioritization, identify a minimal set of features that your early users can test and evaluate. Incorporate engaging techniques, such as the 'buy a feature' method, to keep users interested and actively participating in the prioritization process. In this approach, you provide your users with a virtual budget and present them with a list of potential features. Users can then "purchase" the features they find most valuable within the constraints of their budget.

This interactive exercise can offer profound insights into the perceived value and importance of each planned feature from the customer's perspective. By involving customers in the prioritization process, you can ensure that your product aligns with user needs and expectations, ultimately leading to higher customer satisfaction and product success.

Prioritizing for a mature product

The principles of prioritization remain consistent throughout the product life cycle, from MVP to maturity. Always use the business strategy, product goals, or product vision to select target customers and users, then use their goals and activities to choose the features. However, when your product reaches maturity, you'll have a substantial user base and limited resources, making prioritization even more critical.

As you consider adding new features, evaluate the value of existing features to ensure your product doesn't become bloated and negatively impact usability. Strive to maintain an optimal set of features by not only adding new ones but also removing, replacing, or merging those that no longer provide significant value to users. This approach will help you keep your product focused, relevant, and user-friendly, ultimately leading to sustained customer satisfaction and product success.

In conclusion, effectively prioritizing product features is essential for maintaining customer satisfaction and achieving product success. As you progress through the product life cycle, maintaining a balance between adding new features and refining existing ones is crucial for sustained innovation.



Prioritizing product features is a continual process that demands data-driven decisions, clear alignment with business goals, and active involvement of customers, ensuring a balance between innovation and refinement throughout the product's life cycle.

Establish a design system

"Design is not just what it looks like and feels like. Design is how it works."
- Steve Jobs

Establishing a design system is a crucial step in creating a digital product, as it helps ensure consistency and improves productivity of the developers. A design system combines various elements like style guides, pattern libraries, brand guidelines, and user experience. By using a design system, you can make the product visually appealing and user-friendly while embracing your brand values.

For instance, Airbnb's design system, called Design Language System (DLS), has helped the company establish a consistent and recognizable brand. It includes guidelines for typography, color palette, and UI elements, among others, that are used across all their products. This has led to an improved user experience and a stronger brand identity.

Another example is Google's Material Design system, which provides a comprehensive set of design guidelines for all their products. This system includes design principles, UI patterns, and components that are used across all their platforms. This approach has helped Google create a unified look and feel across all their products, making it easier for users to navigate and use them.

Creating a design system can be challenging, as it requires collaboration between people with different expertise. One approach is to start with research, studying existing brand guidelines to understand the values of your organization and translate them into visual artifacts. The design system needs to be accepted by everyone involved, and feedback should be sought from designers, developers, and other stakeholders.


You may come across two types of elements when creating a design system: tangible and non-tangible. Tangible elements are those that can be seen and used in the application, such as UI elements, patterns, design tokens, and guidelines. Non-tangible elements, on the other hand, are more abstract and difficult to define, such as brand values and team values.

To establish a comprehensive design system, it's important to include the following essential elements:

- **Style Guide:** The style guide serves as the foundation for graphic elements like typography, colors, and transitions. It outlines how these elements should appear to the user and where they should be implemented in the application.
- **Pattern Library:** The pattern library provides a set of functional components that can be reused across the application. This helps to maintain consistency in the design and development process.
- **Brand Guidelines:** Brand guidelines cover theming, accessibility, and user experience. It's important to ensure that the user interface is consistent and intuitive across platforms while embracing the unique benefits of each platform.
- **Design Tokens:** Design tokens are fundamental building blocks that are not tied to a specific platform or technology. For example, instead of defining colors in a platform-specific way, you can use a token like "primary.color" with a specific value. This ensures consistency across the application, regardless of the underlying technology.

By including these elements in your design system, you can improve productivity, maintain consistency, and ensure that your product reflects your brand's values and principles.

Remember that a design system is a living set of artifacts and practices that need continuous improvement, so plan to keep it continuously updated and tracked in your source control.



Establishing a design system is the creative foundation that ensures consistent, user-friendly, and brand-aligned digital products, bringing your organization's values and principles to life through tangible and non-tangible elements.

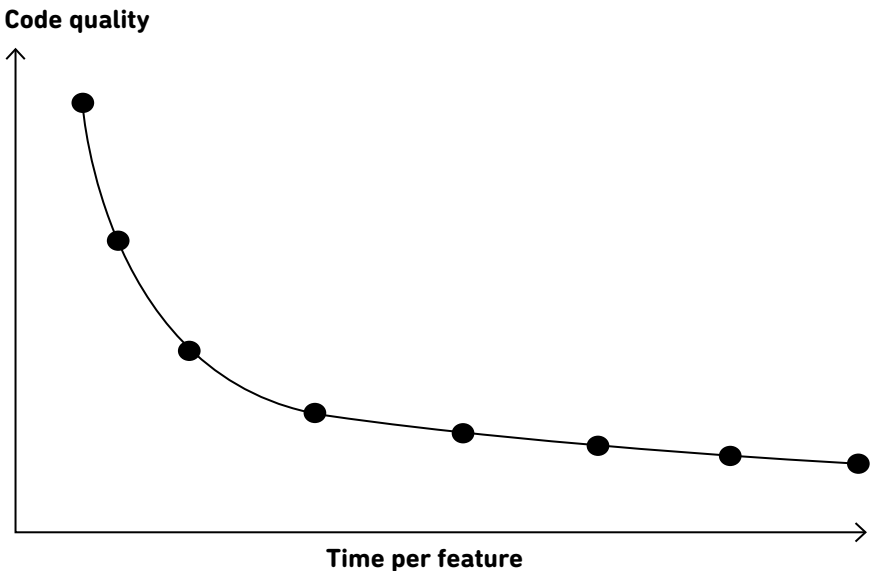
Take control of code quality

“Real quality means making sure people are proud of the code they write, that they are involved and taking it personally.”

- Linus Torvalds

Investing in code quality is essential, as the code you produce directly impacts the product’s user experience and its perception in the market. A simple bug that makes the product unstable can cause users to lose trust, leading to increased churn. As Linus Torvalds once said, “Real quality means making sure people are proud of the code they write, that they are involved and taking it personally.”

Code quality should not be an afterthought for your team. Initiate the process to achieve code quality before writing the first line of code and continue throughout the product’s lifetime. When creating a code quality process, keep in mind the lean principle of “quality at the source” that encourages implementing code quality checks at each step to make it a responsibility of everyone who contributes. Maintaining a high code quality not only improves the product experience but also enhances the productivity of the team as shown below.



Good code has a few distinguishing attributes:

1. **Readability:** Easy to understand and consistent.
2. **Testability:** Can be tested.
3. **Maintainability:** Easy to modify and reuse.
4. **Extensibility:** Code is open to catering to future demands.

Below is a step-by-step guide for you to follow in setting up a code quality practice:

Establish product-specific standards: The first step in setting up a code quality process is to establish a set of standards tailored for the product under development. These standards should guide the development team in writing clean, efficient, and maintainable code.

Implement a peer review process: Introduce a peer review process that evaluates every critical code change according to the agreed-upon standards. Encourage all team members, not just senior ones, to participate in the review process. This approach fosters collective responsibility and accelerates the growth of junior members. Utilize tools like GitHub, Crucible, and Review Board to streamline the review process and make it developer friendly.

Integrate code review tools with task board: Integrating code review tools with your task board makes the code quality process visible to everyone on the team. This integration allows for easy tracking and follow-up on code review findings through tasks.


Use IDE-integrated tools: Manual reviews can miss code style violations. Using tools integrated with developer IDEs can provide instant feedback on code style violations and offer intelligent guidance on best practices. Linters in IDEs offer real-time feedback, while server-based tools such as SonarCloud, VeraCode, SNYK, and Whitesource establish quality gates in continuous integration to prevent violations.

Leverage automated tools: Automated tools are crucial for maintaining code quality over time. Continuously improve the automated rule set to

reduce manual effort, and incorporate common mistakes identified during manual reviews into the automated rule set.

Invest time in unit tests: Unit tests help ensure code robustness and boost developers' confidence when making changes. Writing unit tests helps to improve your code structure and unit designs. Consider employing proven testing techniques such as Test-Driven Development (TDD) or Behavior-Driven Development (BDD). These methods help verify that the code performs its intended function while ensuring minimal, readable, decoupled, and modular code.

In conclusion, taking control of code quality is an essential step in ensuring a successful product that is well-received in the market. By adhering to the steps outlined in this chapter, you can establish a dependable code quality process that will benefit your product. Now that we've addressed the importance of code quality, let's move on to the implementation of an effective test automation strategy, which will serve to further enhance your product's reliability and overall performance.



Prioritize code quality from the outset, involving the entire team in maintaining clean, readable, and robust code, as it forms the foundation of user trust and product excellence.

Embrace test automation

"Testing leads to failure, and failure leads to understanding."

- Burt Rutan

Software testing is a critical aspect of software development, but it can be a resource-intensive process that demands high effort from the team. Further, repetitive testing increases human errors in the testing process. One approach to address this is through test automation that improves accuracy, consistency, and efficiency. Test automation allows the frequency of test cycles to be increased without a corresponding increase in manual effort.

It is impractical to automate all test cases. We need to be selective in test automation as some tests can be expensive to automate and maintain over time. Here are some criteria you may use to select the best candidates for automation:

- Smoke-tests, which test the basic functionality of the system, such as logging in or creating an account.
- Business-critical functionality and workflows, such as payment processing or user authentication.
- Test cases that are time-consuming or difficult to perform manually, such as load testing or stress testing.
- Repeatedly executed test cases such as regression tests, which are run to ensure that existing functionality has not been affected by new changes.

Let us now look at the steps to introduce test automation to your product.

Test tool selection

Select a tool that is suitable for testing the application type, such as web, desktop, or mobile. For example, if you are automating end-to-end tests for web applications, the tool must support headless testing for increased efficiency. For mobile testing, you could use a test tool that can integrate with a device farm. In addition, support for reporting and metrics is another important criterion when choosing your tool. For example, some tools like Selenium WebDriver or Playwright offer robust reporting mechanisms that can help identify issues and improve the quality of your tests.

Define the scope of coverage

Correctly identifying the areas of the application to automate is critical. You also need to identify the type of test performed such as unit, integration, smoke, and regression. For instance, if you are testing a website, you might want to consider often neglected areas such as cross-browser testing, language support, or accessibility depending on your business priorities.

Handling test data

Tests often require input data, and in most test cases, you must generate and input test data. For example, if you are testing a registration form, you might need to create multiple user accounts with different information to ensure that the form is working as expected. You should generate test data while considering data anonymization requirements. You should also think about how you will clean up your data after test execution. Implement a strategy for test data isolation between successive tests. You may create a sandbox environment for your test suites either by cleaning the test data after execution, or through techniques such as in-memory databases or provisioning containerized infrastructure.


Test execution

Setup your triggers on automated tests based on code commits and pull requests. In executing tests, you may require setting up machine and environment configurations to run tests. Some tests such as load-tests may require parallel test execution. Most cloud infrastructures provide support for this. Since you have already isolated your test suite data, use parallelism to save test execution time. Configure distribution of detailed test reports to relevant stakeholders for monitoring and quality control purposes. For example, you could use tools like Jenkins or GitLab CI/CD to automate the test execution and report generation process.

Maintenance

Once you have sufficient coverage of tests, then the team should set aside time to maintain them on a continuous basis. Efforts must be made to keep the test execution time to an acceptable level. Automated tests should be everyone's responsibility, and a mentality should be established to continuously maintain them as changes are made to the System Under Test (SUT). Consider having the automated tests as a quality gate in the pull/merge request process to solve this problem.

In conclusion, test automation can significantly improve the efficiency, accuracy, and consistency of your software testing process. However, test automation is not a one-time solution. It requires ongoing maintenance, and automated tests should be everyone's responsibility. In the next chapter, we will discuss how to shift-left your quality gate reviews, which is another approach that can help streamline your testing process and catch issues early in the development cycle.



Embrace test automation to enhance testing efficiency, accuracy, and consistency, but remember that it's an ongoing process requiring continuous maintenance and collective responsibility from the team.

Shift-left your quality gates

"No matter how good the team or how efficient the methodology, if we're not solving the right problem, the project fails."

– Woody Williams

In today's fast-paced world, customers expect frequent updates and improvements to digital products. To keep up with this demand, many product teams have adopted rapid deployment strategies, with some even deploying multiple times a day. However, ensuring quality in these deployments can be challenging. That's why it's important to have checkpoints or gates in place to assess readiness and identify defects early on. By catching and addressing issues sooner, you can reduce the cost of fixing them later and maintain a higher level of user satisfaction.

Shifting quality gates earlier in the software delivery lifecycle, known as "shift-left," has become increasingly important with the rise of frequent updates and production deployments. By assessing and identifying defects earlier in the development process, the cost of fixing them is significantly reduced. Microsoft is a prime example of a company that successfully implemented a shift-left approach by integrating quality gates earlier in their software delivery lifecycle. As a result, Microsoft was able to catch defects earlier, improve customer satisfaction, and maintain their competitive edge in the industry.

What is a Quality Gate? It's a checkpoint to validate any stage of your software delivery life cycle. For example, validating a Pull Request with an automated build before merging to a stable branch can be considered a quality gate.

Quality Gates for requirements

It's crucial to identify defects as early as possible, even during the requirement gathering phase. Apply the following backlog review gates when grooming requirements. By doing so, you can detect issues in your problem statement before proceeding to latter phases.

1. Define clear acceptance criteria or definition of done for user stories. This ensures all team members have a clear understanding of the requirements.

2. Describe each user-story in many test scenarios as possible. This further elaborates the requirements, providing more implementation details.
3. When adding new features to existing products, cross-check new requirements with existing telemetry data to verify the validity of the new feature.

What are Quality Gates during development and testing?

During development and testing, it's crucial to have several quality gates in place to ensure code quality and overall product quality are maintained. To achieve this:

1. Utilize Continuous Integration (CI) to compile the code for each commit made by developers. This ensures a consistent code state, allowing team members to collaborate on the project without breaking each other's work.
2. Implement unit and integration tests within the automated build pipelines to verify that the code is functioning as expected.
3. Enforce quality standards at pull requests. To preserve a stable branch, you can test every pull request by deploying it to a staging environment and by automatically executing tests.
4. Validate each commit using static code analysis. This ensures code quality and reduces technical debt.
5. Scan each code commit for security vulnerabilities during continuous integration builds. During software development, sensitive information, such as passwords, may be inadvertently added to your code. Automated scanning tools can identify these cases and alert you to potential security vulnerabilities.


Quality Gates in deployment and pre-release stages

Implementing the following quality gates during deployment and pre-release stages helps you detect and address issues before they reach production:

1. Set up a pre-production or staging environment that mirrors the production configuration. This enables you to test the software product under conditions identical to the production environment.

2. Use the staging environment to conduct vulnerability and penetration tests before release. This helps you identify and address security issues before they can impact your production environment.
3. Employ Canary Deployments to validate business scenarios with a subset of end-users. Canary deployment is an effective strategy that involves releasing new software features to a small group of users initially, allowing you to test and validate the functionality in a real-world setting.
4. Test your configuration management mechanisms that are in place. When provisioning your infrastructure using automated templates, consider executing a dry run or manually verifying the changes before deploying them to production.

As you establish quality gates throughout your development lifecycle, it is also vital to pay attention to security aspects, which will be the focus of the next chapter. By doing so, you can further safeguard your product and maintain a strong security posture in an ever-evolving digital landscape.



Shift-left your quality gates to detect and address issues early, maintaining product quality in an era of rapid deployments, and ensuring customer satisfaction through proactive quality assurance.

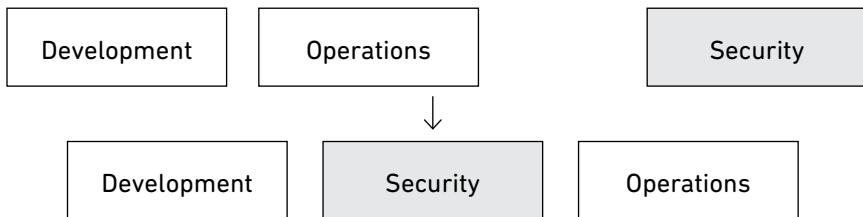
Establish a SecOps practice

“The only truly secure system is powered off, cast in a concrete block, and sealed in a lead-lined room with armed guards.”

-Gene Spafford

Cutting corners on security could lead to severe consequences or regulatory penalties down the line. Introducing SecOps, a collaborative effort between security professionals and the development/operations team, helps integrate security into your operations to ensure your product's safety.

SecOps maturity requires having implemented processes, feedback loops, and security compliance right in the release pipelines. The result is fewer production security fire-fighting incidents that drag developers into emergency remediation. This leads to higher velocity, faster time-to-market, and a credible product.



SecOps introduces security at each phase of the software development life cycle, such as requirements analysis, design, coding, and deployment. This mindset is also known as “Security by design.”

Your product is not the only source of vulnerability, it's only one part of the equation. Let's look at some other areas that you shouldn't neglect to avoid hefty penalties.

External components

It is important to identify and categorize vulnerabilities to understand how they are introduced into your system. A digital system is usually a collection of internally developed components and externally sourced components, such as third-party libraries or code packages.


Mature external libraries or components are usually more reliable. When introducing an external component, consider the reputation of the vendor, active maintenance, the size of the user community, and known vulnerabilities in addition to functional suitability. However, keep in mind that popularity could also introduce threats, as popular libraries are often targeted by hackers. Stay vigilant about relevant vulnerability announcements to mitigate these risks.

Deployment environments

The deployment environment and configuration are crucial security concerns. To avoid common mistakes during this stage, consider adopting a “secure by default” approach by permitting the least privilege by default for any activity. Best practices include closing unnecessary ports, making virtual networks private, and opting for Platform-as-a-Service offerings wherever possible.

Remember that your developers may not be experienced security experts. Enlist the assistance of security experts to help you as appropriate. A great way to enhance your team's security knowledge is connecting them with external communities. For example, OWASP (Open Web Application Security Project) offers a regularly updated set of resources that highlights security concerns for web application security, focusing on the most critical risks.

In conclusion, achieving SecOps maturity involves implementing appropriate processes, establishing fast feedback loops, and integrating security compliance directly into the release pipelines. As a result, production security fire-fighting incidents are reduced, preventing developers from being pulled into emergency remediation. This approach leads to improved DevOps efficiency, increased velocity, accelerated time-to-market, and a more credible product.



Embrace SecOps to seamlessly integrate security into your product's lifecycle, fostering a safer, more credible, and efficient development process.

Protect personal data

“Privacy forms the basis of our freedom. You have to have moments of reserve, reflection, intimacy, and solitude,”

- Dr. Ann Cavoukian

We live in an age where protecting your digital self is as important as protecting yourself physically. With the rise of the data-based economy, data is now one of the most valuable assets. Giants like Facebook and Google thrive on monetizing personal data. Many companies that have breached their users' trust have faced large penalties from regulators. For example, Google was fined \$57 million under GDPR for failing to provide transparent information to users and obtain valid consent.

Data privacy should not be confused with data security. Data privacy is the right of an individual for their data to be kept secure. Data privacy governs what data is collected, how it is collected, stored, used, managed, and shared. Data security, on the other hand, is focused on protecting data from being compromised. However, data privacy becomes meaningless if there is no data security. In fact, data security and data privacy complement each other when it comes to personal data protection.

“Privacy means people know what they're signing up for, in plain language, and repeatedly. I believe people are smart. Some people want to share more than other people do. Ask them.”

- Steve Jobs

Almost any product you build would collect user data to a certain degree. The simplest form being a user's name and email address. The more personal and sensitive the data, the more important it is for the user to be aware that their data is being collected, its purpose, and to provide consent towards its use.

What privacy laws are applicable to your product?

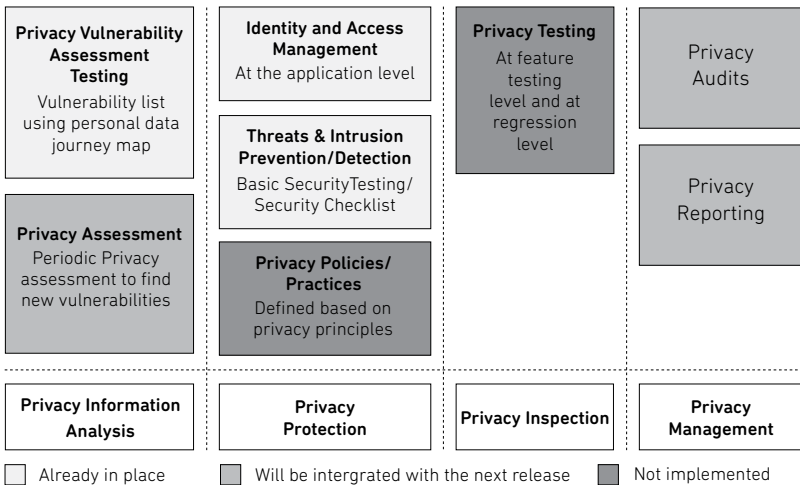
If the product you are building deals with personal data, you might need to adhere to one or more applicable privacy regulations. Sometimes the liability stemming from these regulations may not be obvious. For example, you may be developing a product for an Independent Software Vendor (ISV), who sells it to a reseller, who in turn sells it to an enterprise

that stores end-user details. You may not get a glimpse of any customer data. However, you may still be held accountable for any privacy breaches under specific regulations where you are treated as a data sub-processor.

Data privacy regulations are usually applied based on the geographic location of the data subject. Data subject refers to any individual who can be identified directly or indirectly based on the collected data. Some of the popular privacy regulations are the European Union’s General Data Protection Regulation (GDPR), California Consumer Privacy Act, Canada’s Consumer Privacy Protection Act (CPPA), and Singapore’s Personal Data Protection Act (PDPA). You may need some professional help in figuring out what laws and regulations apply to you. Most of these regulations are vague, with grey areas open to interpretation. Once you have that understanding covered, defining a privacy architecture is an ideal place to begin your data privacy journey.

Defining a privacy architecture

Privacy architecture defines a framework for processes, policies, and technologies necessary to protect personal data. The following is a template that you could use to create a privacy architecture:



* Source: Developed in-house and in use in several projects. Based on <https://security-and-privacy-reference-architecture.readthedocs.io/en/latest/09-privacy-principles.html>

This diagram depicts a swim lane approach to analyze, implement, test, and manage privacy. Each swim lane comprises one or more activities that must be performed periodically. A simple color code is used to define the progress in each area.

Privacy Information Analysis

To develop a privacy architecture, you should start by running a privacy assessment. A privacy vulnerability assessment will help identify how personal data flows through your application and can be used to create a vulnerability list that identifies all possible risks. Periodic privacy assessments will help you find new vulnerabilities and fix existing ones.

Privacy Protection

Privacy protection is the process of safeguarding data once information is captured. Identity and Access Management (IAM) is an essential component of privacy protection, as it defines how you implement authentication and authorization at an application level. Measures should be taken to protect data in transit and at rest. Additionally, separation of development, testing, staging, and production environments are crucial components of privacy protection. Threat and Intrusion Detection is another crucial aspect of privacy protection, as it helps detect and mitigate any unauthorized access to personal data.

Privacy Inspection

Privacy Inspection refers to the process of testing for privacy at the feature and regression level. It involves verifying that all personal data collection, storage, processing, and sharing activities are compliant with relevant privacy laws and regulations. At the feature level, Privacy Inspection is conducted during the development process to ensure that privacy controls are incorporated into each feature. At the regression level, Privacy Inspection is conducted after each release to verify that privacy controls remain effective and that no new privacy vulnerabilities have been introduced.

Privacy Management

Privacy Management involves defining the necessary steps to ensure the process implementation and sustainability. Privacy Audits are periodic reviews to ensure that the team follows applicable laws and regulations.

These audits can be conducted internally or by third-party auditors and help identify potential privacy risks and areas for improvement. Privacy Reporting involves creating reports that document the status of privacy management within an organization. These reports can help organizations understand their level of compliance with privacy regulations, identify areas for improvement, and track progress over time.

In summary, data privacy is not a topic that can be overlooked or relegated to the periphery in favor of more visible commitments, as a data privacy breach can have significant consequences. By prioritizing data privacy and integrating it into every aspect of their operations, your product can not only avoid negative consequences but also build trust and credibility with your customers.



Prioritize data privacy by integrating it into every aspect of your operations, avoiding consequences, and building trust and credibility with your customers.

Setup your support processes

"The goal as a company is to have customer service that is not just the best but legendary."

- Sam Walton

Establishing effective customer support processes is vital to the success of any digital product. It involves developing a clear understanding of customer needs, and then creating systems and processes to ensure those needs are met in a timely and satisfactory manner.

One of the most important aspects of customer support is responsiveness. Customers expect prompt and helpful responses to their inquiries, whether they come via email, chat, or social media. To meet this need, you need to establish clear response time standards and ensure that your support teams are adequately staffed.

For example, companies like Zappos have set the gold standard in customer support by making it a priority to respond to all customer inquiries within a matter of hours, if not minutes.

Empathy is also a key component of effective customer support. Customers want to feel like their concerns are being heard and that they are being treated as individuals. This means that support representatives need to be trained in active listening and effective communication skills, and that they can respond to customer concerns in a compassionate and respectful manner.

Another important element of customer support is transparency. Customers want to know what's going on behind the scenes, and they want to be kept informed about any issues that may affect their experience with a product or service. This means providing clear and timely updates about things like service outages, product recalls, or changes to terms of service.

Below are some useful tips for you to build a best-in-class customer support process.

Use right tooling

To provide fast, expert service with every interaction, your support team needs the right tools at their disposal. One example of a company that does this well is Shopify. Shopify's support team uses a variety of tools to manage customer inquiries, including shared inboxes, help desk software, and automated workflows.

Your customers shouldn't have to wait in long queues and the process of handling customer requests can be automated with artificial intelligence services such as Automatic Speech Recognition (ASR), Natural Language Understanding, Text to speech providing Human-like interactions. The customer support requests will then be automatically recorded as tickets in your help desk system or Customer Relationship Management (CRM) system with their urgency level and priority level.

Provide self-service support

In addition to providing direct customer support, it's important to offer self-service options to customers. This not only frees up your support team's time but also empowers customers to solve their own problems. Skilled agents can still help in-person with more complex questions and requests. Self-service frees up the time of your agents to focus on the needs where human intervention is needed.

Hire right people

Of course, none of this is possible without a great support team. When hiring customer support professionals, look for traits like attentiveness, patience, communication skills, and empathy. To provide top-notch support, your team needs to be well-trained and empowered to make decisions. There's a difference between the time taken to respond and the speed in which the problem is resolved. Encourage your team to respond to customers promptly, but don't rush them to close requests before a customer's issues are entirely resolved.

Balance automation and personal support

While AI-powered chatbots can be useful for providing quick answers to simple questions, they are no substitute for personalized, human support. It's important to strike a balance between automation and personalized service. An example of a company that does this well is HubSpot. HubSpot's chatbot, named "Conversational Marketing," is designed to help visitors find what they need quickly, while also providing a way to connect with a human representative if necessary.

Using data and respond proactively

There are two main approaches to providing customer service: reactive support and proactive support. Reactive support is the more traditional approach, where the customer contacts the support team when they encounter an issue. However, proactive support has proven to be more effective in building customer loyalty. The way to implement proactive support is by using data to pre-identify issues and alert customers based on their usage patterns. By analyzing customer data, companies can identify potential issues and proactively reach out to customers to help before they even encounter the problem.

In conclusion, providing excellent customer support is critical to the success of your product business. By following the best practices outlined above and learning from the examples of successful companies, you can build a support team that is responsive, empathetic, and effective at meeting your customers' needs.



Building a successful digital product requires creating a customer support process that prioritizes responsiveness, empathy, transparency, and a balanced approach to automation, ultimately leading to customer loyalty.

Delight customers at onboarding

"A poor onboarding experience is hard to come back from and is the fastest way to lose a customer."

- Paul Philip

The onboarding process is a critical phase in the user journey that sets the tone for the relationship between the user and the product. An effective onboarding strategy considers users' goals and how they intend to use the product. It not only teaches users how to use the product but also helps them realize the immediate value of the product. A good onboarding process keeps users engaged, improves conversions, and creates a seamless experience for the user.

Many tech companies have invested in their onboarding process, such as Slack, which uses a setup wizard to guide users through the product, and Dropbox, which offers a product tour to highlight key features. By delivering a great onboarding experience, products can increase user engagement, improve customer satisfaction, and ultimately, drive business growth.

One company excelling in user onboarding is Airtable. Airtable is a cloud-based collaboration platform that allows users to create and share databases, spreadsheets, and other productivity tools. Their onboarding experience is designed to make it easy for users to get started and understand the product's key features. When a user signs up, they are taken through a series of interactive tutorials that highlight different aspects of the platform. Airtable also offers customizable templates that allow users to quickly create and customize their own databases. The onboarding process highlights Airtable's unique features (such as "Blocks" feature, which allows users to add custom functionality to their databases). These unique features are highlighted in the onboarding process, and users are encouraged to experiment to see what works best for them.

An effective onboarding process is crucial in ensuring user engagement and retention. The following are some key steps in a typical onboarding process:

Sign up: Make the sign-up process simple and straightforward by asking for only the minimum required information. You could also offer easy sign-up options with third-party services like Google, Slack, or Facebook.

Welcome email: Send a personalized welcome email to your users after they sign up, expressing your gratitude and letting them know you value them. Provide resources like product tour videos, help center articles, or FAQs to help them get started.

First login: Once the user logs in for the first time, it's essential to show them around and guide them in using your product. Offer a welcome pop-up or a set-up wizard to help them get started quickly. Provide a product walkthrough to help them understand the key features and functionalities of your product.

Personalization: Providing a personalized onboarding experience can make the user feel more valued and increase their engagement with the product. For example, Netflix recommends content based on a user's demography and preferences, making the onboarding experience more personalized and engaging.

Integrations, invitations, and data imports: Allow users to set up integrations, invite colleagues, and import data, but don't force them to do it immediately. For example, Asana allows users to invite their colleagues and import data from other tools, making it easier for them to collaborate and get started quickly.

Offer support: Provide easy channels for obtaining support and resolving any issues they may face during the onboarding process. By following these steps, you can ensure a smooth onboarding experience for your users, increasing the chances of user engagement and retention.

Creating an effective onboarding process is the key to enhancing user engagement and retention, setting the stage for a positive and enduring relationship between users and your product.

Phase 6

Optimize through learnings

| | |
|------------------------------------|-----|
| Leverage data for decision making | 130 |
| Develop your product portfolio | 134 |
| Maintain architectural consistency | 137 |
| Position for a rapid growth | 140 |
| Organize your lead funnel | 142 |
| Establish license management | 144 |
| Build a team brand | 147 |
| Manage cost escalation risks | 159 |

By the time you reach the Optimize phase, you've already validated your product's Unique Value Proposition (UVP) and confirmed your customers' willingness to pay. Your core features have been developed and tested by early customers during the Build phase. Now, the focus is on refining your product to make it even more valuable to your existing users and more attractive to new ones. This phase is also concerned with scalability, as your product is likely to experience exponential growth and face increasingly demanding users. To achieve success, you need to ensure technical consistency, optimize pricing models, and rely on growth hacking to drive adoption.

Leverage data for decision making

"Data is the new oil. It's valuable, but if unrefined, it cannot really be used."
- Clive Humby

By now, your software product is out there on the market. As you gain traction in the market, it's important to make decision based on the usage patterns and feedback from customers, rather than relying on the product team's opinions and experiences. In this chapter, we will discuss the benefits of a data-driven approach to feature prioritization and explore some effective ways to collect and analyze data.

Re-visit your feature prioritization process

Data-driven feature prioritization involves collecting data on user behavior, usage patterns, and feedback to identify which features are most valued by customers. This approach allows you to make informed decisions about which features to prioritize based on actual customer needs, rather than on assumptions or guesses.

In data-driven decision making, you should develop a set of hypotheses and test them to see whether they are supported by the data. For example, you might hypothesize that a self-booking feature will reduce the number of calls to the call center. You can then collect data to see whether your hypothesis is supported by the data. If the hypothesis is confirmed, you can ensure that you have developed a value-added feature that meets customer needs. To collect data, you can use tools such as heatmaps, mouse tracking tools, and product analysis applications, or ask questions directly from a selected set of users.

Q1: What is the advantage of having a self-booking feature?

Hypothesis 1: Increasing the usage of the application will decrease the number of calls to the call center. Measurements:

- Avg. call center volume weekly
- Avg. calls for appointment bookings
- Number of appointments through the online booking system

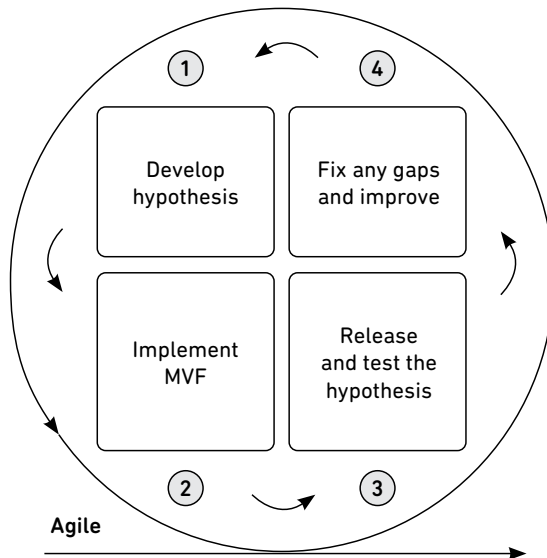
Hypothesis 2: If users can manage their appointments themselves, that will result in a decrease of the cancel and change appointment requests to the call center. Measurements:

- Avg. cancel requests to call center
- Avg. change appointment requests to call center
- Number of cancel requests from the online application
- Number of change requests from the online application

Shortening the feedback loop

To get the most benefit from the data-driven approach, it's important to shorten the feedback loop. The concept of MVP can be applied to feature level too. That means releasing a Minimum Viable Feature (MVF) to the end-user and using user experience research methodologies such as A/B testing to gather data to fulfill your hypothesis. You can use different feature revisions with feature toggling to verify which one gives more value to the user and eliminate guesswork.

The following diagram shows a methodology for data-driven decision making while practicing agile software development.



To apply data-driven decision making and to optimize every feature of the product through MVFs, you can follow these steps:

- **Identify the feature:** Select the feature that you want to optimize and define the goals and objectives that you want to achieve with it. This will help you develop a hypothesis and determine the metrics that you need to track.
- **Develop a hypothesis:** Develop a hypothesis about the feature based on your understanding of your customers and their needs. The hypothesis should be testable, measurable, and relevant to your business goals.
- **Implement an MVF:** Implement a Minimum Viable Feature (MVF) that includes the minimum functionality required to test the hypothesis. This allows you to get feedback from users quickly and at a lower cost than building a full-featured product.
- **Verify the hypothesis:** Collect data to verify the hypothesis, including user behavior data, feedback, and user satisfaction metrics. Analyze the data collected and compare it with the hypothesis. Determine if the data supports the hypothesis or if there are gaps that need to be addressed.
- **Pivot and iterate:** Identify changes that need to be made to the feature to improve its usefulness. Make changes to the MVF and prioritize new features based on the insights you have gained from the data. Repeat the process with new features or updates to existing features.

For example, a SaaS company may want to improve their task management feature. They could develop a hypothesis that adding deadline functionality to the task management feature will increase user productivity. They could implement an MVF that includes only the basic deadline functionality and collect user behavior data, feedback, and user satisfaction metrics to verify the hypothesis. If the data supports the hypothesis, they could fully implement it. If the data does not support the hypothesis, they could re-prioritize the backlog and develop a new hypothesis to test.

In conclusion, a data-driven approach to prioritization is crucial to ensure that your product meets the needs of your customers. By collecting data on user behavior, usage patterns, and feedback, you can make informed decisions about which features to prioritize and implement. Validating hypotheses, shortening the feedback loop using an MVF and user experience research methodologies such as A/B testing can further improve the value of the product to your users.



Embrace a data-driven approach to feature prioritization, collecting insights from user behavior, and shorten feedback loops to continually refine your product based on real customer needs and usage.

Develop your product portfolio

“Don’t find customers for your products; find products for your customers.”

- Seth Godin

Effective product portfolio management involves overseeing all aspects of your products to optimize return on investment and ensure sustainable business growth. In this chapter, we will explore key considerations for building a diverse product portfolio, as well as strategies for blending your offerings.

Diversifying your product portfolio

Product diversification involves expanding your product into new markets and should only be considered once you are stable and established in your primary market. Diversification can lead to new business opportunities and mitigate risks in the event of an industry or market segment downturn. Diversification can also be used to defend against competitors and capture market share.

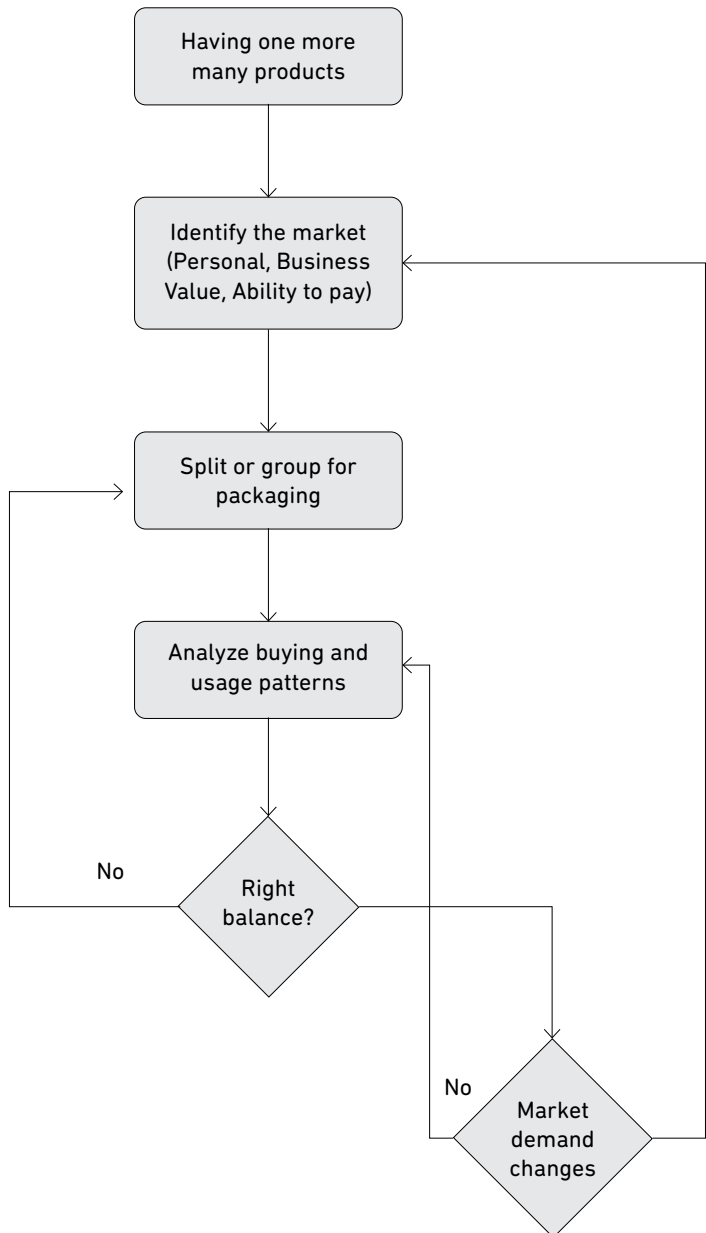
Software companies have the potential for exponential growth, as they are not bound by geographical and logistical limitations. Therefore, consider diversification to find profitable niche markets for your offerings.

Blend Your Offerings

To grow your reach through a blended offering, you must first conduct preliminary research to find the sweet spot. This can involve techniques such as splitting and grouping products, repackaging, and repricing. The right balance depends on the overall vision and objectives of your organization as well as the objectives of your users.

The following diagram summarizes how to approach the continuous improvement of your product portfolio blend.

The right balance depends on the overall vision and objectives of your organization as well as the objectives of your users. Until that is found, you can iteratively improve your offerings based on market feedback data. Now, let’s focus on a few of the above tactics that will help you successfully blend your product portfolio.



Repackage, Rename and Reprice

To make your product more appealing in different markets, consider altering the way you present it. For example, you can repackage, rename, or reprice your product to cater to different demographics.

Offer Complimentary Products

Complimentary products add value to one another. In other words, they become better when consumed together, like a good movie and popcorn. This will help your company to increase customer loyalty and lock-in. The Apple ecosystem is a great example of complementary products.

Splitting Products

Sometimes you may split your product, looking at how users value and consume it. This allows a product to be specialized and offered independently to access new markets.

Offer choice through customization

Allowing users to customize your product based on their specific requirements can be a powerful tool for attracting and retaining customers. However, be careful not to overwhelm users with too many choices. Remember, building and maintaining an effective product portfolio is an ongoing process. Continuously improving your offerings based on market feedback and data can help you stay competitive and grow your product business.



Craft a diverse product portfolio strategically, considering product diversification for growth and blending offerings to create value, continuously adapting based on market feedback and data.

Maintain architectural consistency

"Good engineering is characterized by gradual, stepwise refinement of products that yields increased performance under given constraints and with given resources."

- Niklaus Wirth

Software product businesses rely on technology to deliver value to their customers. As business strategies change, it's critical to ensure that technology adapts to these changes. This means that architects must work closely with product managers and developers to keep the product architecture consistent as the business evolves. Failing to do so can have serious consequences, ranging from minor annoyances to complete product failures.

To illustrate the impact of architectural inconsistencies, consider the case of a successful software product company that failed to manage their architecture effectively. The company's product was widely used in the market, but as they added new features and functionality, the architecture became increasingly complex and drifted from its original design. This led to brittle code, reduced performance, and security issues. Eventually, the product became outdated and lost market share to competitors with more modern and efficient systems. To avoid these issues, it's important to understand that architecture requires continuous effort and maintenance throughout the product's lifespan. While the initial blueprint is laid out at the inception, the architecture must evolve with the product to ensure it remains consistent and efficient.

What causes architectural inconsistencies?

Architecture inconsistencies broadly fall into two categories, technical and non-technical. Technical causes are those stemming directly from dealing with the design and implementation of the architecture. Non-technical causes are those stemming from organizational and staffing considerations.

Technical Causes

- **Big design upfront:** Attempting to plan out the entire architecture upfront without being open to changes can lead to a lack of flexibility to meet future market needs. Good architects tend to defer architectural decisions until necessary information is available.

- **Poor architecture governance:** Violation of the rules, conventions, and processes associated with the intended architecture can prevent the product from meeting requirements. For example, if a layered architecture only communicates between adjacent layers, violations can occur when the development team bypasses adjacent layers to directly communicate with non-adjacent layers.
- **Incorporation of third-party components:** Integrating readily available software components can introduce architectural style incompatibilities that create new challenges.

Non-Technical Causes

- **Time to market pressures:** The pressure to quickly release a product can compromise the architectural integrity.
- **Poor communication and knowledge transfers:** Inadequate communication of design to the development team can lead to improper implementation of the architecture. Poor management of knowledge transfers can also lead to erosion of the architecture, especially with high staff turnover.
- **Lax engineering processes:** Some organizations maintain a more rigorous approach to prevent or mitigate architecture-related challenges. The more rigorous an organization's approach, the less likely inconsistencies will occur.

Balance intention and emergence

Traditional architecture approaches such as Big-Design-Up-Front lead to extensive early architecture work. This can create copious documentation and unvalidated decisions. An alternative approach is to follow agile architecture practices which is a combination of intentionality and emergent design, where architecture continuously evolves as the team learns more.

- **Intentional architecture:** Defines a set of purposeful, planned architectural strategies and provides guidance for inter-team synchronization. For instance, a software development team may decide to use an agile architecture approach when developing a

new product. They could start with an intentional architecture that defines the overall structure of the application, including the components, their interactions, and the data flow. This architecture could be communicated to the development team to ensure a shared understanding of the system.

- **Emergent architecture:** This is an evolutionary and incremental approach to architecture. Emergence helps to respond to changing user needs, minimize waste and allows the architecture to grow with time. For instance, the previously discussed could use an emergent architecture approach to refine and adjust the initial architecture as development progresses. They may discover that some components need to be split or combined, or that new components need to be added to support additional functionality. By continually evaluating the architecture and making incremental adjustments, the team can ensure that the architecture is responsive to changing needs and requirements.

Overall, balancing intentionality and emergence through architectural agility is a powerful way to approach software architecture in an agile and responsive manner. By using intentional architecture to provide guidance and emergent architecture to refine and adjust the system, software development teams can ensure that their products are adaptable, flexible, and responsive to changing market needs. Agile architecture also fosters a DevOps culture by ensuring solutions are architected for continuous delivery to shorten the time to market.

Maintain architectural consistency by balancing intention and emergence, addressing both technical and non-technical causes of inconsistencies, and fostering agile architecture practices to ensure adaptability and responsiveness.

Position for a rapid growth

"Think big, start small, then scale or fail fast."

– Mats Lederhausen

In today's highly competitive market, digital products need to constantly grow their customer base while managing the cost of customer acquisition. Promotions and growth hacking are two popular techniques that product companies use to achieve this goal. In general, promotions take your message to the market and position the product to your target audience. In contrast, growth hacking is using radical, aggressive approaches to reach a larger audience without incurring the high costs of traditional approaches.

Promotions

Promotional activities aim to take your message to the market and position your product to your target audience. Promotional activity includes advertising, offers, public relations (PR), media placements and in-person marketing. You can create a calendar of activities along with the channels, budget and partners that will complement your sales targets. To develop an effective promotional strategy, businesses need to:

- Decide on a budget for promotions that aligns with their sales targets.
- Understand the strengths and weaknesses of various types of promotional messages for their target audience and decide what works best.
- Identify the best mix of promotional channels for their audience, such as advertising, social media, and public relations.
- Determine the conversion strategies for customer leads that should be in place as part of the overall promotional process. This ensures that all leads are properly captured, qualified, nurtured, and harvested.

Growth hacking

Growth hacking is an aggressive approach to reach a larger audience without incurring the high costs of traditional approaches. It involves experimenting with many different marketing channels quickly to find ways to grow the business as cost-effectively as possible. Growth hacking

is commonly used by startups as a cost-effective alternative to rapidly expand in the early phases. Take the example of Dropbox, the online storage and file-sharing company struggled to sign people up in the early stages. They needed a way to onboard people at a lower customer acquisition cost. They opted for a referral system that offered their existing customers free space for every successful referral. As a result, existing users began sending out emails and doing free marketing for Dropbox.

Measuring the effectiveness of your marketing activities

This topic would not be complete without a discussion on how you can measure the results of your efforts and the business impact. This will give you objective means to optimize your marketing spend. A few indicative metrics are given below.

- **Return on Investment (ROI):** Calculate and compare the sales revenue a campaign brings on every dollar spent.
- Average cost per sale.
- **Cost per lead:** Used to calculate the cost-effectiveness of marketing channels.
- **Lead funnel:** The conversion rates at each stage of the lead's journey and other behavioral information reveal the bottlenecks of the conversion process.

By visualizing how their marketing activities perform, you can improve them in the long run. This is where lead funnels come into the picture. A lead funnel provides product teams with a visual representation of the customer journey, helping them identify areas for improvement and optimize their marketing strategies.

Supercharge your digital product's growth with strategic promotions and innovative growth hacking, all while measuring your success through ROI and lead funnels for sustained expansion.

Organize your lead funnel

“You have to generate revenue as efficiently as possible. And to do that, you must create a data driven sales culture. Data trumps intuition.”

– Dave Elkington

Peter Drucker’s words “You cannot manage what you do not measure” are as true today as when they were first recorded. You cannot measure what you don’t track, and whatever you don’t track, you will also lose over time.

The amount of effort you put into tracking your lead funnel will depend on the type of product you are offering. For example, if you have a B2C product with a low-ticket value, you may have many leads progressing through the funnel, requiring less manual attention per lead. On the other hand, if you offer a B2B software product for enterprises with a higher ticket value, the focus on each individual lead and addressing their specific concerns would be much higher.

It’s not practical to track your leads using pen and paper. To effectively manage your lead funnel, you need to invest in an integrated Customer Relationship Management (CRM) tool that allows you to track your leads from the first point of contact to the end of the sales cycle. This tool should offer you the ability to manage contacts, run marketing campaigns, manage a deals pipeline, and provide reports on overall progress and effectiveness.

CRM tools can be integrated with the product itself to drive new sales opportunities and up-sales. For example, a click on a subscriptions page or attempting to access a locked feature can trigger an alert to a sales consultant.

Stages in your lead funnel:

The lead funnel consists of multiple stages where lead is evaluated before entering a subsequent stage. By associating a probability with each stage, you will be able to make sales forecasts based on the pipeline and the time to win a deal.



Marketing Qualified Leads (MQL) are leads that have shown an interest in your product and moved from being just aware of it to being ready to make a purchase decision. The objective at this stage is to nurture each lead and move them closer to a purchase decision by showing more engagement and interest (e.g., opening your emails, joining webinars, signing up for a demo, asking for more information).

Once a lead shows enough interest, they become Sales Qualified Leads (SQL) and should be treated as such. At this stage, they are handed over to the sales team for contracting.

The final two stages of the funnel involve maximizing your revenue opportunities with your existing customers. The first stage is the Upsell stage, where satisfied customers become prospects for up sales. The last stage is the Referral stage, where happy customers refer others to your product.



In conclusion, managing your lead funnel is a critical aspect of customer acquisition. By investing in an integrated CRM tool, understanding the stages of the funnel, and measuring its effectiveness, you can optimize your lead funnel and maximize your revenue opportunities.

Mastering your lead funnel paves the path to acquisition success, where every stage is an opportunity to nurture and flourish, maximizing your product's potential and your business's growth.

Establish license management

"There are two kinds of companies, those that work to try to charge more and those that work to charge less. We will be the second."

- Jeff Bezos

Licensing is the method of exercising your pricing strategy. Poor licensing management can lead to poor sales, revenue leakage, operational inefficiencies, and user dissatisfaction. You have already read about pricing strategies, let's see how licensing models impact how organizations and individuals purchase your products. It is necessary to provide customers with transparent information to make an informed decision, without being surprised later by hidden costs. Given the recent evolution of software licensing models, this is even true for B2C customers.

Today, licensing is split across pay-upfront models and pay-as-you-go (PAYG) models or a hybrid of these. Traditional pay-upfront licensing models often results in a higher barrier of entry for customers. In contrast, PAYG models allow users to have flexibility in their cost by letting them choose their commitments for either a defined period or level of usage. Gartner indicates that due to the widespread adoption of software-as-a-service (SaaS), all new entrants and 80% of established vendors offer PAYG business models today.

Pay-upfront licensing models

Let's look at some pay-upfront licensing models:

- **Perpetual license:** A widely used licensing model where a digital product is sold once, and the licensee can then use the software forever. (e.g., Microsoft Windows license)
- **Floating license:** It allows you to share a maximum number of license instances/seats among a group of people. This can be offered either as a simultaneous usage cap or allocated to specific individuals. A whitelist license is a variation of the floating license model that allows access only to a pre-approved, specific set of users.
- **Project-based license:** This supports collaboration between multiple users who work on a given project, where the license is taken for a specific project.

- **On-demand corporate license:** This combines aspects of other licensing methods to create a frame licensing agreement that allows you to add or remove users at an incremental cost, on demand. This provides flexibility for the product vendor, especially when dealing with large enterprises.
- **Support and maintenance license:** A commitment towards the continuity of the product that includes maintenance and support. Typically, these licenses carry service level agreements (SLAs) and related penalty clauses for mission-critical services.
- **Academic License:** An academic license is not a distinct licensing model. Rather, it is provided to a specific audience. It is a widely used model to capture future markets by creating loyalty. e.g., student license for Microsoft Office software.

Pay-as-you-go (PAYG) licensing models

Pay-as-you-go models yield recurring payments from your userbase that enables continuous cash in-flows. Let's have a look at common PAYG models.

- **Subscription license:** This is a popular model, where the end-user licenses the product repeatedly for a defined period. It could be per user, per group of users, or even per concurrent user. (e.g., online magazines, medium.com)
- **Use-Time license:** Defined by the time duration the user is given access to the product (e.g., Compute service on Cloud platforms).
- **Feature license:** Used to limit the number of features available in a product. E.g., The product licensing tier advances as the user demands more sophisticated features from the product.
- **Trial license:** Allows prospective users to try out the product before purchase by enabling it for a fixed duration or with a limited set of features.
- **Freemium license:** Similar to a trial license, you offer users basic or limited features at no cost and then charge a premium for advanced features or product support. This can attract a large set of initial users with a low financial commitment. While these users may not purchase upgrades at once, you can still collect data, insights and even earn ad

revenue. If the freemium model is done poorly, you will end up with many free users who never convert to paid users!

- **Credit-Based Subscription:** Users pay the price to the subscription account for redeemable credits. Like a gift card, credits are applied to users' accounts to help cover costs until they are exhausted or expire. E.g., audible.com credits, AWS/Azure credits.

Remember that the licensing model should evolve continuously to serve market changes and counter competitive offerings. The preferred license model needs to be flexible, configurable, and support discounts and refunds.

Once we select the preferred licensing model, the next challenge is to implement this mechanism in your product. Sophisticated licensing models require technology provisions such as feature toggling, metering, traceability, license management systems, and payment gateway integration built into the product itself. These capabilities need to be provisioned in your product architecture.

A good licensing solution must be convenient for both users and administrators. You must decide whether to build your own licensing solution or integrate it as an off-the-shelf tool. It is recommended to consider proven licensing management software unless your business case justifies internal development.



Effective licensing models empower your product's pricing strategy, ensuring transparency, flexibility, and sustainable growth for both your organization and users.

Build a team brand

"Individual commitment to a group effort—that is what makes a team work, a company work, a society work, a civilization work."

- Vince Lombardi

If you want to build a winning digital product, you need to have a power team to back you up. Having a solid brand for that team is just as important as having a killer product brand. Why? Well, because it shows customers and industry colleagues that your team knows their stuff and can be trusted. Having a strong team brand also makes your team feel more connected and prouder of their work, which leads to better quality work and a more positive workplace. So how do you go about branding your team?

Well, first things first, you should figure out your team's personality and values. Once you've got that down, you can create a team brand statement that reflects who you are and what you stand for. But that's not all. You must get the word out there! One way to do that is by sharing your team's journey through a blog or social media channel. Tell the story of how your team came to be, share interesting learnings, and give a behind-the-scenes look at what it takes to build a great product.

Another way to establish your team brand is by presenting at meetups, conferences, and other industry events. By sharing your team's knowledge and expertise, you can position yourselves as thought leaders in your field and gain credibility among your peers.

And don't forget to encourage your team members to build their own personal brand too! Whether it's by releasing open-source projects or tools, writing blog posts, or speaking at events, having team members who are experts in their respective fields only strengthens your team brand.

Here are three examples and case studies of successful team branding initiatives:


Slack: The team behind the communication tool Slack has successfully established a strong team brand by creating a blog and social media presence that shares the company's culture, values, and work practices.

The company's blog features articles written by different team members, sharing their experiences and insights on various topics related to the product development journey. This has helped to create a personal connection with their customers and humanize the brand.

Intercom: The customer messaging platform Intercom has built a reputation for being an expert in the field of customer support and communication. The company's team members regularly share their knowledge and expertise by speaking at conferences, writing blogs, and offering consultations to the industry. This has helped to establish the company's credibility and thought leadership in the industry.

HubSpot: HubSpot has created a strong team brand by placing a strong emphasis on continuous learning and personal development. The company offers its employees access to a wide range of training programs, mentorship opportunities, and resources that help to improve their skills and knowledge. This has helped to create a culture of continuous improvement and innovation within the company.

In conclusion, building a strong team brand is crucial for creating a successful product development team. By establishing a team personality and values, sharing knowledge and expertise, creating a continuous learning culture, and inspiring team members to build their personal brand, a team can increase its credibility, attract top talent, and foster a sense of belonging and pride. Actively using appropriate marketing channels to take your team's credibility to the outside world is an important step towards creating a followership within your community.



A strong team brand not only builds trust and connection with customers but also fosters excellence, credibility, and personal growth, making it a cornerstone of successful product development.

Manage cost escalation risks

"There is no compression algorithm for experience."

- Andy Jassy

Your business activities and technology decisions are hopefully stable now. As such, it makes sense to relook at all aspects of your business to optimize costs while retaining the end value to your users. Costs directly translate into the bottom line, therefore reducing your expenses will positively impact on your product's financial viability. By now there should be less pressure from a time-to-market aspect, and thus you should have adequate time to focus on operational excellence.

Operational costs

Most businesses gain immediate cost savings from reducing operational costs. This comes in many forms, and one of the common ones is the infrastructure costs. In previous phases, you may have explored multiple options to get the technology to work. Some of the decisions you made may not be the most cost-effective choices. There could also be some deployment environments that are still running, but no longer required.

Cloud service providers today offer multiple technology options with different pricing models, and it's worthwhile to investigate all available choices. For example, if you have any standalone virtual machines in your application, moving some of these into shared containers would make sense from a cost perspective. Another example would be to move some of the infrequently used data into cold storage, which is cheaper.

You can also look at optimizing your business processes. The whole philosophy of lean manufacturing is something that could be explored here. The processes you have formed during the inception of the business may be discrete and suboptimal by now. Back then, business priorities would have been different. But now is the time to reconsider new priorities and refine your processes to make them lean, eliminating waste.

Optimize your people costs

A large portion of your R&D cost is your people. A post-pandemic reality is the realization that skilled talent can collaborate and build great products,

regardless of where they are located. In a world scrambling for talent to support digitalization, you are no longer constrained by geographical boundaries to source people.

A rational decision most product vendors take is to outsource non-business critical operations, while retaining the critical domain and technical knowledge in-house. Outsourcing is generally cheaper and enhances your operational processes while improving quality due to specialization.

You should note that it is crucial to retain your internal employees. Replacing resources is generally more expensive than retaining them. Also, knowledge leakage and transition overheads are detrimental to the organization. As such, it is important to look after your core employees while outsourcing non-key activities of the business.

Reduce costs with automation

Another option to save costs is by automating activities that are repetitive and time consuming to perform manually. For example, you may be creating reports manually at the end of the month for decision making. This is a good candidate to automate, as it repeats monthly and is typically prone to human errors.

Deployments are another area that you can automate. Modern DevOps platforms easily allow you to automate code and infrastructure delivery to make the whole deployment repeatable. We have also discussed test automation in multiple places in this book, which is a good candidate to automate. This will help you save time and money while increasing testing quality as we eliminate any human errors.

Most companies today use Robotic process automation (RPA) that emulates humans' actions interacting with digital systems. RPA bots can understand what's on a screen, complete the right keystrokes, and perform actions to accomplish goals much more efficiently than humans. You should look for potential RPA candidates among systems that don't offer standard ways of integration but are connected within a process. For example, you may find RPA candidates among customer support, lead funnel management, etc.

Reduce costs of external dependencies

If you are tightly coupled with a 3rd party library, service, or platform, they may have an unfair influence on your business. The Epic games vs. Apple battle back in 2021 is a good example for this. Since Apple controls the application marketplace, Epic games had a little to no choice than accepting the terms and services of Apple's platform.

So, it also makes sense to re-examine these dependencies and evaluate the long-term risks to your business. Investments to mitigate these risks might be expensive in the short run, but it makes your business far more stable in the long run.

Despite all these cost-saving efforts, you should never compromise the product's value, reduce customer experience or hurt your employees. The focus should always be on optimizing activities and eliminating waste.



Optimizing costs without compromising value, customer experience, or employee well-being is key to enhancing your product's financial viability and long-term stability.

Phase 7

Harvest your return

| | |
|--------------------------------------|-----|
| Nurture your leads | 154 |
| Embrace industry standards | 156 |
| Prepare for surge volumes | 158 |
| Tackle production issues promptly | 161 |
| Harness the power of inbound content | 164 |
| Embrace referrals and upsales | 167 |
| Prepare for contingencies | 169 |

Now that your product is successful in the market, you will aim to maximize profit. This is where you make significant returns to pay back your investors and build reserves for your future ventures. You will aim to retain happy and loyal customers, up-sell more, and get referrals to grow your business.

In this phase you will explore low-cost growth mechanisms to optimize your marketing expenses, implement contingency and risk mitigation measures to continue enjoying the highest returns.

Nurture your leads

"The best marketing doesn't feel like marketing."

- Tom Fishburne

No matter what product or service you offer, it's unlikely that your customers will always make an impulse purchase. Instead, they'll go through the stages of awareness, consideration, and decision-making before deciding to buy from you. Lead nurturing involves the marketing and promotional activities you do to educate and engage your potential customers throughout this process. As a simple analogy, it is like the trust and relationship building that takes place during dating, a process that ideally leads to a greater commitment!

Let's say you run an online fitness coaching SaaS product. A potential customer might come across your website through a Google search or a social media post. They may not be ready to commit to buying a coaching package right away, but they're interested in learning more about how your product may help them to improve fitness. So, they sign up for your email newsletter or follow you on social media to keep in touch. Now it is your lead nurturing process that must take responsibility for converting this prospect into a paying customer. To nurture this lead, you need to communicate with them in a planned and deliberate way. Important activities of this process involve:

Finding touchpoints

Depending on your audience, you might use channels like email, social media, or direct messaging to engage with your leads. For example, you might send out a weekly email newsletter with fitness tips and motivational stories, or post daily workout challenges on Instagram. The goal is to stay top of mind and build a relationship with your potential customers. Most electronic channels offer advanced mechanisms such as 'remarketing' that allows you to engage your audience repeatedly, even though you may not know who they are.

Creating relevant content

You'll need to identify the specific content that resonates with your audience and share it in a structured way. For example, you might start by sharing general fitness tips and advice, and then gradually introduce more specific content tailored to your potential customers' needs as you learn

more. Continue to maintain engagement by providing useful content such as relevant fitness research papers and new trends to gauge their interest. Once the lead is interested, then build your credibility by demonstrating how you have solved the problem for others. In attempting to convert, you could offer free workout plans, nutrition guides, or testimonials from happy customers who have achieved their fitness goals with your help.

Finding the right frequency and triggers

No one likes to feel 'spammed' just because they have engaged with you. Do not overwhelm your potential customers with too much communication, but also don't let them forget about you. Experiment with different frequencies and triggers to see what works best for your audience. For example, you might send out a weekly newsletter, but also follow up with a special offer or discount code if a potential customer hasn't engaged with your content in a while. If your CRM system supports it, automate prospect-initiated activities to trigger an appropriate response workflow.

As you nurture your leads, it's important to track your campaign outcomes and analyze the results. This will help you make informed decisions about which channels and content types are most effective, and how to optimize your marketing spend. You might look at metrics like open rates, click-through rates, conversion rates, and customer acquisition costs to see what's working and what's not. With a little bit of experimentation and analysis, you can create a lead nurturing strategy that turns potential customers into loyal fans of your product.



Lead nurturing, akin to building trust in a relationship, converts potential customers into loyal advocates by engaging them through relevant content, finding the right balance of communication, and analyzing results.

Embrace industry standards

"I believe good regulation may hurt Facebook's business in the near term but it will be better for everyone, including us, over the long term."

- Mark Zuckerberg

Building trust with users is crucial for any software product, and one effective way to do this is by adhering to industry standards. Compliance with these standards may even be mandatory in certain regulated domains, such as healthcare and finance. Compliance in a software product context means having a recognized and trusted third-party certify that your product and processes meet a specific standard. For example, if your software development process is ISO 27001 certified, it should comply with industry best practices on information security.

Let's explore the importance of compliance and regulatory standards in more detail, and how they can benefit your product.

Competitive advantage

Suppose a potential customer comparing payment processing products—one complies with PCI DSS (Payment Card Industry Data Security Standard), while your product doesn't. Which platform would they trust? Compliance standards like these can give you a competitive advantage and boost your brand image. Compliance standards are even more important in a B2B setting, where products are formally evaluated objectively against each other before a purchase decision is made. By complying with standards, you'll build trust with users and differentiate yourself from competitors who may not be compliant.

Regulatory requirements

Regulatory requirements are another important consideration, especially if you're in a domain like Healthtech or Fintech. Some regulatory requirements are mandatory, and failure to comply can lead to legal issues. For example, GDPR may be applicable depending on the geography of your customers. It's important to be aware of these dependencies from the start of your product journey.

Another example illustrating the importance of compliance standards in the healthcare industry. Suppose you're developing a mobile app that allows patients to track their health information and communicate with their doctors. To ensure compliance, you must follow the standards set by regulatory bodies such as HIPAA, which mandate strict requirements for handling patient data.

Assess compliance across your components

If you're using third-party components or cloud platforms like Azure and AWS, it's crucial to evaluate their compliance levels as well. The compliance of these platforms and components can limit the level of compliance you can offer to your users. For instance, if you're using a virtual machine in the cloud for a Fintech application, the cloud service provider should be PCI DSS compliant in managing physical and virtual infrastructure. Apart from the platform, you also have a shared responsibility with your team to ensure product level compliance.

Adopting industry standards can increase your product's value and drive the quality of the entire industry upwards as adoption increases. Passenger safety in the automobile industry has improved over the years through the implementation of standards. The same concept applies to software products, and compliance standards enable you to implement it systematically.



Embracing industry standards not only builds trust and competitiveness but also ensures compliance with regulatory requirements, safeguarding your product's reputation and enhancing the entire industry's quality.

Prepare for surge volumes

"In times of rapid change, experience could be your worst enemy."

- J. Paul Getty

By now, your product has reached its end users and the customer base may also be growing steadily. Depending on your domain and user based, you may experience random or seasonal surge volumes. From a technical point of view, this is where you must stretch the product's capabilities to scale.

Your architecture should establish a strategy to cater to this kind of rise in demand. You can utilize the power and elasticity of the modern cloud to scale the product on the fly. By default, the system resources can be kept low to save cost. This is ideal, especially at the initial stage as you cannot predict the level of demand and growth. When your application demands more power, modern cloud providers give you the ability to scale up and scale out. Scale up means increasing the capacity of your existing resources, while scale out means introducing more resources to handle the increased load.

Tackle seasonal demands

Some applications tend to have seasonal surges in demand. For instance, a payroll platform would see the bulk of its activity towards the end of a month, or a productivity app like Google Sheets might have most of its traffic during working hours.

The recommended approach is to design an architecture that can utilize the total available power without significant fluctuations. This might mean decoupling systems, delaying processing to a time where system load is minimal, or processing the load using finite resources by taking more time. For instance, you can design the system to process heavy tasks like preparing invoices or analytical data processing during off-peak hours. Beyond this, modern cloud architectures also make room for auto-scaling. This gives the ability to either scale up or scale out for a brief time when there is a heavy workload. The system would automatically revert to defaults when the load reduces.

Modularized scaling

It is also necessary to understand the bottlenecks in your application. It could be that your entire application doesn't need to be scaled up or scaled out but may just be the database or the web tier which needs extra capacity. Robust architectures use proper layering and loosely coupled components which allow you to scale independently. For example, the database layer can be easily scaled up to a higher performance tier without upgrading the web servers.

Geographically distributed scaling

It is also worthwhile investigating the geographic regions where your traffic comes from. Modern cloud providers give you the ability to relocate or scale out resources closer to your end customers. For example, if most of your customers connect from the United States and a few from Australia, your server capacity can be proportionately allocated to reflect this demand. Modern cloud service providers give you the ability to do this with minimal effort as the demand changes.

Respect your service level agreements

Be aware of performance thresholds specified in your service level agreements. As you commit to your customers, remember to factor in the throughput variations of your cloud service provider.

The 'Signal' messaging app provides a perfect example of not being able to grab a once-in-a-lifetime opportunity. Elon Musk endorsed Signal in January 2021 in response to WhatsApp's updated privacy policy, which raised concerns about user data privacy. Musk tweeted "Use Signal" to his millions of followers, sparking a surge in downloads for the messaging app. However, Signal's servers were not prepared for the sudden surge in demand, this resulted in a missed opportunity for Signal to capitalize on the market, as many users who tried the app during the surge likely had a poor user experience and may have turned to other messaging apps. If they had a strategy to scale when required, they wouldn't have disappointed all their new users.

In conclusion, surge volumes are an inevitable part of scaling your product. As your user base grows, your architecture should be designed to handle sudden and seasonal surges in demand. By preparing your architecture to handle surge volumes, you can ensure that your users have a positive experience and avoid missing out on opportunities for growth.



To thrive in a dynamic digital landscape, prepare your product for surge volumes by building scalable architectures, embracing modularized and geographically distributed scaling, and respecting service level agreements.

Tackle production issues promptly

"Make every interaction count. Even the small ones. They are all relevant."

- Shep Hyken

Despite having tight quality controls in place, you won't eliminate all production defects. There will always be issues that are unknown to you at this time, but they will surface in future. As such, you should be able to provide fast fixes to ensure that your product is functioning.

You must have proper issue reporting and an escalation process established to identify them. Once a defect is raised in production, you should do triage and determine the severity and impact. Thereafter, you should fix it while ensuring the stability of the product.

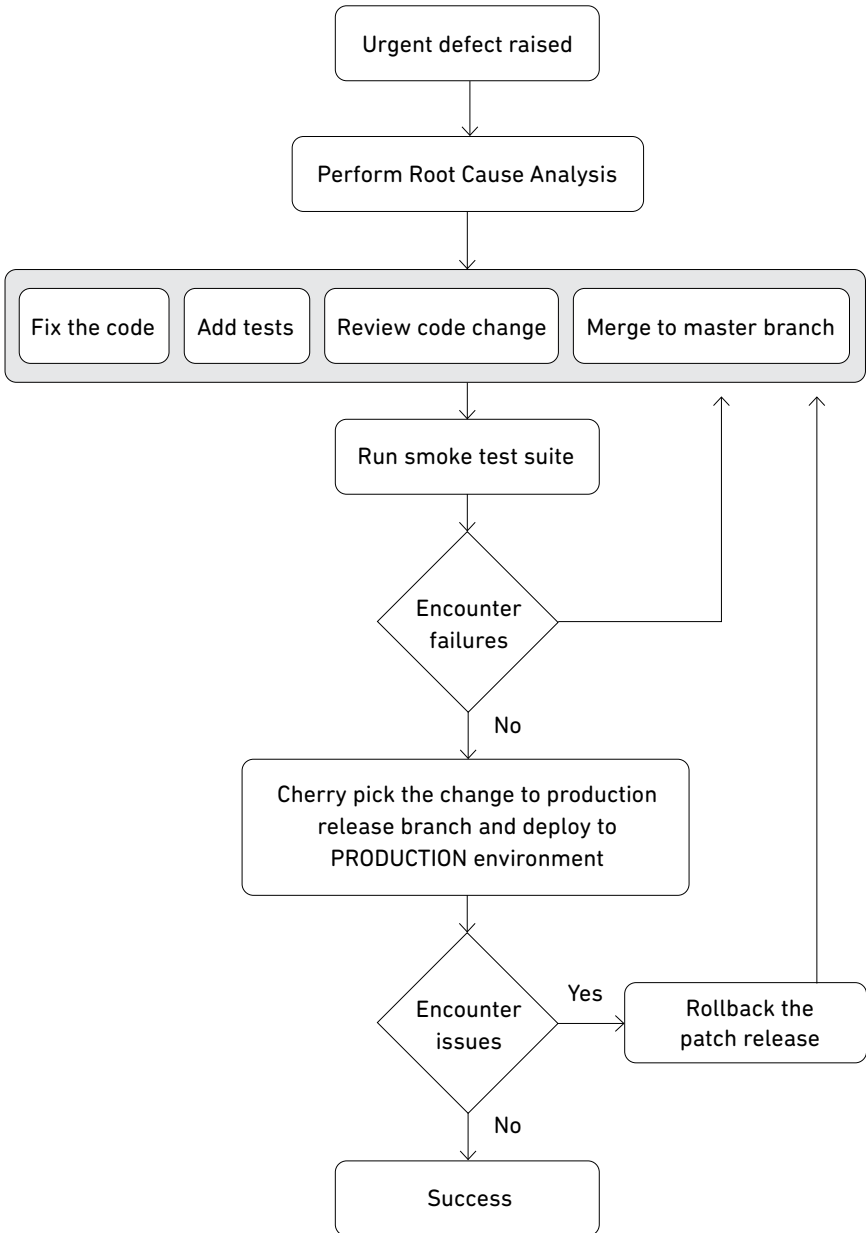
You may need development and DevOps processes with automation to support fast fixes. Such procedures may include having Continuous Integration and Continuous Deployment (CI/CD), a branching strategy and a release strategy in place.


Branching strategy

Let's elaborate more on why adopting a branching strategy like 'trunk-based development' is essential. With trunk-based development, the fix you need to bring into production will first be added to the development branch (also known as the 'trunk'). Then, you can choose (cherry-pick) these changes from development and merge them into the production release branch. This is illustrated in the diagram below.

When your fix is deployed to production, ensure it's already there in all the subsequent releases. On the contrary, if you merge changes to the production release branch first, there's a possibility of missing out on the subsequent merger with the development branch. Your customers will not be happy to see a critical defect reappear after they see it being fixed.

You should always take sufficient time to verify the fix and its impact no matter how urgent the fix is. The last thing you want to do is to introduce additional defects into production while performing a fast fix.





Swiftly addressing production issues with proper reporting, triage, and fixes, along with a robust branching strategy like trunk-based development, ensures product stability and customer satisfaction.

Harness the power of inbound content

"Content is fire, social media is gasoline."

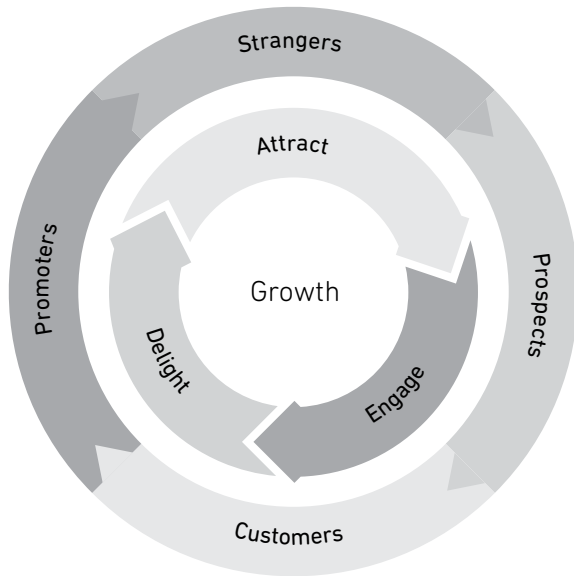
- Jay Baer

Inbound marketing helps potential customers to find your product through valuable content and experiences. This process takes place much earlier than the actual purchase decision. As a result, you can build a preference for your brand generating more revenue.

Inbound marketing engages potential customers through organic means, such as search engine optimization and sharing links among friends. An example of this is when a product or brand is subtly mentioned in a relevant blog post. Such techniques grab the attention of a potential customer more effectively than when an advertisement annoyingly pops up in the middle of a video.

HubSpot, one trusted platform for inbound marketing, suggests a three-stage approach for your content strategy.

1. **Attract:** Drawing potential customers through valuable content and conversations that position the company as a trusted advisor.
2. **Engage:** Presenting insights and solutions to address their pain-points and aspirations to purchase your product.
3. **Delight:** Assisting the customer to achieve their outcomes. When customers share their success, they become promoters driving more users to your product.



HubSpot – Flywheel Model

Attracting strategies

You can create blogs, videos and social media posts to reach your audience. Also search engine optimization (SEO) can be used to drive search traffic to your product to attract potential buyers in the long run.

When publishing the content as above, specific words and phrases related to the product can be included on your website to attract more people. This will allow the pages of your organization to organically appear on the search engine results page (SERP).

Engaging Strategies

Aim to establish a long-term relationship with your users by selling a solution rather than selling a product. Your engagement activities must be focused towards increasing the lifetime value of your customers.


Although email is not the most modern engagement strategy, you can still derive higher rates of engagement by making your content stand out from regular email formats.

Delighting strategies

This is to make sure that your customers stay satisfied and supported. You must obtain measurable feedback of your users' experience to drive continuous improvement. You can use chatbots and surveys to obtain feedback at scale.

At the same time, monitoring social platforms also plays a significant role in understanding customer sentiments. Customers frequently use social media to express their thoughts and experiences, ask questions and provide feedback. Responding to their interactions in a prompt manner shows that the organization listens and cares for them.

You may encounter situations where despite your best efforts, some customers may not find your product a mutual fit. In such instances, make every effort to part-ways on a positive note.



Inbound marketing, with its three-stage approach of attracting, engaging, and delighting customers through valuable content and experiences, builds brand preference and generates revenue.

Embrace referrals and upsales

"The best advertising is done by satisfied customers."

- Philip Kotler

Have you experienced your online shopping site suggesting goods outside of your shopping list? How many times have you ended up adding them to your cart or referred these products to your friends? Such suggestions are the result of a seller's upselling strategy.

What are upsales?

Research suggests that the probability of selling to existing customers is high as much as 60-70%, while the chance of selling to a totally new prospect is only about 5-20%. Upselling is a sales technique to sell more products and services to existing customers. It is used to encourage customers to upgrade to a more expensive version, buy repeatedly or to buy complementary offerings.

Done right, upsales can build stronger relationships with customers. Upgraded or better versions of a product or service benefits both parties. This builds a deeper relationship with the customers, and they may stay with the company for a longer time.

Upsales increases customer lifetime value (CLV), which is the total revenue contribution a customer makes over time.

What are referrals?

A referral is where your existing customers introduce you to new business. People naturally trust recommendations from fellow purchasers over vendor's promises.

Referral marketing has many forms. At times your customer refers you to another potential buyer one-on-one or may endorse you publicly on a B2B referral site. Referral marketing done via social media platforms is also known as influencer marketing. These influencers can be your friends, celebrities, domain experts, or online reviewers. It would be good to identify who might be the right influencers for your product as their role is becoming increasingly important.

Your referral marketing plan should identify the market segment, offer frequency, budgets, benefits and communication strategy. The following are a few considerations when communicating your referral program to your customers.

Choose the right time: Ask for a referral at a time you make your customer happy with your services.

Make the referral offer simple and clear: Your offer should be straightforward to grab the attraction of the referee. For example, if someone refers to a new customer, they are entitled to a \$100 reduction on their next invoice.

Upsales and referrals are two great techniques you can also use to boost your growth. Make use of every opportunity to activate a referral in as many workflows of your product.



Embrace the power of referrals and upsales to not only boost revenue but also build stronger customer relationships, increasing customer lifetime value.

Prepare for contingencies

"In preparing for battle I have always found that plans are useless, but planning is indispensable."

- Dwight D. Eisenhower

Your product operates in a dynamic environment. There will always be forces beyond your control. These include changes in legislation, adverse market and economic conditions, or compelling events such as the following.

Change in Industry Regulations: Changes in industry regulations can have a significant impact on SaaS product companies. For example, the recent changes in regulations governing online gambling in various countries have forced SaaS companies providing gambling software to adapt their products to comply with new regulations.

Economic Recession: Economic recessions can impact product companies in several ways. Product companies may see a reduction in customer demand, longer sales cycles, and a decrease in venture capital funding. For example, during the 2008 financial crisis, many SaaS startups went bankrupt or were forced to merge with larger companies to survive.

Security Breaches: Security breaches can have severe consequences for product companies. In addition to the loss of customer data and reputation damage, companies may face lawsuits and regulatory fines. For example, in 2013, Adobe experienced a massive security breach that exposed the personal information of millions of users. The breach led to a class-action lawsuit against Adobe and a significant loss of customer trust.

Global Pandemics: The COVID-19 pandemic had a significant impact on the product industry, with many businesses having to adapt to remote working and virtual events. SaaS companies that provided solutions for remote working, video conferencing, and e-commerce saw a surge in demand, while those that were not prepared for the sudden shift in market conditions struggled.

These forces would directly influence your operations and impact business continuity. This should be addressed by disaster recovery and contingency planning strategies. Consider the following as you plan your contingency response.

- Research and gather knowledge on external factors like market conditions and competition to assess the risks that would affect your business.
- Conduct a comprehensive risk assessment. Identify and prioritize your risks based on its 'likelihood of occurrence' and 'severity of impact' on your business.
- Thereafter, create a plan to address these risks by defining the roles and responsibilities for risk mitigation.
- Prepare a contingency budget, as additional funding may be required to address these unforeseeable events.

An example of contingency planning can be seen in the development of critical systems that require high availability. For instance, if you are building a system that needs to have 99.99% availability, you need to ensure that the product downtime is less than 8.64 seconds per day. This means that you must have redundancy coupled with automatic failover to avoid any single point of failure.

In conclusion, it is crucial to be prepared for contingencies and have a plan in place to address any unforeseen events that could impact on your business. By conducting comprehensive risk assessments, delegating roles and responsibilities, and preparing a contingency budget, you can minimize the impact of these events and ensure business continuity. Remember, if something can go wrong, it will, so always be prepared!

In a dynamic environment, prepare for contingencies by conducting comprehensive risk assessments, delegating roles, and having a contingency budget, ensuring business continuity even in the face of unforeseen events.

Phase 8

Retire voluntarily in time

| | |
|-----------------------------|-----|
| Reinvent your solution | 174 |
| Plan your migration journey | 187 |
| Deal with the residual data | 180 |
| So... what's next? | 182 |

Regardless of how well the product is built and maintained, there's a time to take it off the market. ISVs should be watchful to identify this need to disrupt their business model before a competitor does. Retirement can be brought on by a significant technology shift, competitor entrance, or changes in user-behavior. This means you re-invent the technology platform, business model or user experience to retain your edge in the market. Another scenario to retire a product can come about due to an acquisition or merger, which can involve a superior product.

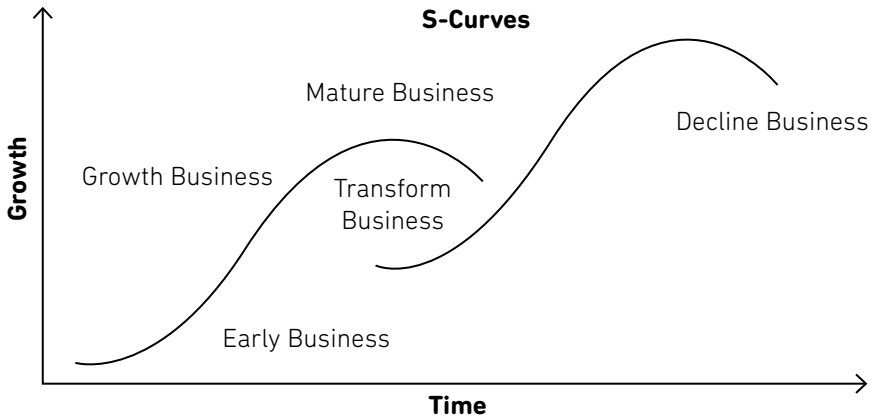
In this phase you should provide clear and pain-free migration paths, coupled with promotions and incentives to motivate customers to migrate. You will ideate the next generation solution, plan for data and user migration and deal with your residual data.

Reinvent your solution

"Innovation is the specific tool of entrepreneurs, the means by which they exploit change as an opportunity for a different business or a different service."

- Peter Drucker

Every software product has a lifetime and someday, it needs to be retired or reinvented. This could be due to various reasons. For instance, there could be an advancement in technology that brings great business benefits yet might not be supported by your architecture. Or it could be due to a new emerging megatrend that does not align with your product. You must start thinking of the next generation of your solution in advance, before the market compels you to retire your product.



But, how do you know when to reinvent? To answer this, we first need to understand the reasons why software products become obsolete.

The S curve of a business

Any business, including software, goes through various phases in its journey. In the early stages of product development, you can welcome changes in the environment as they create good pivot points. However, for a mature product, it would be difficult to keep pace with these changes, indicating the end of its lifetime.

How can you identify these turning points for your product? One way is when the product stagnates and slows growth. You observe a trend where your product no longer can win consistently against a competitor's product offering. You must constantly observe the environment to identify these cues. Let's now look at some of these contributing factors in detail.

What makes a product obsolete?

At a high-level, there are both internal and external factors that make a product obsolete. Let's look at these factors in detail.

External factors

- **Market dynamics:** Software products need to continuously be innovated to keep an edge over the competition. Sometimes it becomes a challenge to modify the product to meet these new demands. For example, new legal and policy enforcements may require a fundamental change in your product.
- **Competition:** Your competitors may come up with an innovation that disrupts your business model. If so, you either need to extend your current product or reinvent the next generation.
- **Trends:** There could be shifts in user behavior that demands a change in your product. For example, an application which does not provide both web and mobile support would not be acceptable today.


Internal Factors

- **Outdated technology:** Use of legacy technologies might lead you to think of the next wave of your product. If not, you could be exposed to serious security and reliability issues. Depending on the domain your product is operating in, there could be new technologies available which allow you to reap significant cost savings.
- **Technical debt:** If the application isn't properly maintained over time, the cost of change may increase, limiting your ability to respond to market demands.

Restarting the journey with more confidence

At some point, you may realize that it is time to reinvent your product. Though the steps are the same, you will be in a better position to build the new product, with experience in your domain and a well-established brand. More importantly, you might already have funds to invest, making the journey much more manageable than when you were a startup.

When you decide to move forward, it's important to plan the transition for your users and communicate as early as possible to set expectations on the new version to come.



Innovation and adaptability are keys to staying relevant; when your product faces stagnation or market shifts, seize the opportunity to reinvent and restart your journey with confidence.

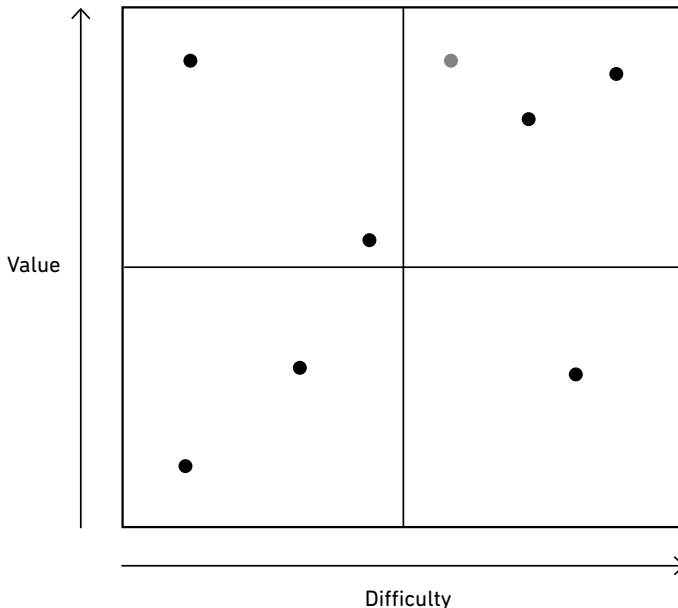
Plan your migration journey

"You can't just ask customers what they want and then try to give that to them. By the time you get it built, they'll want something new."

- Steve Jobs

The first step in your product migration journey is to take an inventory of all available features. Also, you should understand the problematic non-functional areas, such as security and performance. Remember that times have changed since when you first built your product. See what user behavior and market expectations your product must meet to be relevant. You should initiate the activities done during the Explore-Focus-Immerse phases to systematically derive the solution.

This activity should provide you with the list of features your next generation solution should provide. Thereafter, you can plot these features in an x/y chart, where the x-axis represents the difficulty of implementation, and the y-axis represents the value.



Features in the top left quadrant (high value, low difficulty) will be an excellent place to start. However, you can't ignore the top right quadrant. Though these may incur a higher cost to implement, they still have higher value to your users.

Migration approach

There are well-known patterns that can be used in the modernization effort. The most notable being breaking the entire application into small, loosely coupled modules, e.g., microservices. Rather than porting the entire application at once, you can extract features step by step, starting with the high-value, easy-to-implement features.

Most of the time, cloud migration is a part of a larger modernization plan. Rather than performing a full lift-and-shift, restructuring the application to make full use of the cloud will yield more value.

Consider the following aspects in your migration plan.

- **Platform migration plan:** Describes how you would move from the legacy environment to the new environment. You should determine which components from the old system and new system will run in parallel. Then, establish a roadmap how your legacy platform will be phased out. Consider using the old system as a platform to validate the accuracy of the new system.
- **Data migration plan:** Apart from the system components, your product's data needs to be migrated as well. You might have to convert data formats to fit the new data schema. Consider using professional data migration tools to increase reliability.
- **User migration plan:** It is critical to ensure a smooth transition for your end users with minimal impact. There are two migration strategies you can use. A bulk user migration is suitable if you can afford system down time. Consider a phased user migration based on geography or user type etc., to migrate users in batches for a more controlled transition. A phased migration will give the illusion of a zero-downtime migration.

In addition to the above, plan for detailed knowledge transfers covering new technologies, functional improvements, and migration processes across all your team members. This will help your operations to run smoother once the new system is live.

End of life (EOL) support

This describes the final stage of how a product is supported in its lifecycle. You should create an 'end of life' plan and communicate it to your customers well ahead. This plan would indicate the timelines for the following.

- General availability, the product is in active development and available for purchase.
- End of sale, i.e., you would no longer sell to new customers actively.
- End of development, i.e., the team no longer accepts feature requests as the product transitions to a maintenance mode.
- End of maintenance, i.e., the team would not focus on even hot fixes and patches other than critical security updates. Only limited support would be available for the product.
- End of support or service, i.e., the customer accepts all liability by continuing to use the product. In a SaaS environment, this could mean the end of service availability and all forms of support.

You can consider providing incentives to your old users to migrate to your new platform.

Success in the ever-evolving digital landscape requires a proactive approach to migration and end-of-life planning, ensuring seamless transitions, and maintaining customer trust.

Deal with the residual data

"Data is a precious thing and will last longer than the systems themselves."

- Tim Berners-Lee

Would your users be comfortable with their data being stored in applications that are no longer in production? Data protection controls, policies and regulations address this concern. These controls provide your users with the peace of mind that any data that is no longer relevant will be dealt with, according to your consent.

As you retire your product, one of your main concerns should be to dispose of this obsolete data responsibly using data shredding techniques. When retiring your legacy system, certain data might become obsolete. Additionally, various legal aspects often prevent you from keeping data that is no longer required. Just as regulations govern the collection and storage of data, they also specify how data should be disposed of when no longer in use.

Another downside of holding obsolete data is the cost of storage. Especially, if you're using any cloud storage services, you can easily stack up expensive storage bills just to store data without any business value.

Consider the following when shredding data:

a. Returning data to users

Have a provision to return user data before it is permanently removed from the system. You may be legally obliged to do this before retiring the product in certain industries. So, be prepared to provide the functionality granting rights to the data owners to export their content.

b. Data availability timelines

You may be required to keep the data alive for a certain period after retirement due to regulations and to maintain goodwill. Make sure you proactively communicate and specify such terms in your service level agreements (SLA).

c. Maintaining shredding evidence

You could adopt a manual or an automated data shredding procedure.

Either way, it is recommended to keep data deletion evidence (e.g., in logs) for audit purposes. This is important when deleting data upon user request if you need proof of such activity.

d. Hardware-level shredding

Not just the digital forms of content, even hardware needs to be appropriately disposed if they contain sensitive information. Physical media may need to be shredded or crushed to ensure compliance in certain industries.

e. Using a specialist data shredding service

Data shredding too can be outsourced to specialized companies. However, you need to ensure that the selected service provider can prove relevant accreditations, such as EN15713.

You must align your data shredding procedures to widely accepted standards such as NIST 800-88, DoD 5220.22-M, HMG IA Standard 5 or DIN-66399, to ensure data recovery is not possible later.

Data shredding on the cloud

Can you really ensure that data is destroyed in your cloud environment? In the cloud, your data is regularly being backed up and replicated across multiple data centers for availability. Thankfully, most cloud providers handle data destruction automatically based on your actions. However, it is important to review the service provider's contract to ensure it complies with the level of standards you want to adhere to.

In closing, if it is required to keep historical data for analytics, consider removing sensitive information (e.g., personal, medical data) from the raw data prior to storage by anonymizing your data.

Responsible data disposal, aligned with industry standards and regulations, not only safeguards user privacy but also reduces storage costs and ensures compliance throughout a product's lifecycle.

So... what's next?

As we come to the end of this book on building market-winning digital products, it's clear that creating a successful digital product is a sophisticated and multi-faceted process. However, with the right mindset, know-how, and execution, some of you will create a product that users love and is purposeful. The content of this book can serve as a handbook for you to understand the different phases a product goes through and the important strategic focus areas at each phase.

Successful digital products require an organized approach that guides you through the lifecycle journey of understanding the market, focusing on a problem, planning the journey, building and validating, optimizing through learnings, harvesting returns, and finally retiring voluntarily in time. In this book, we have covered each of these phases in detail to help you navigate the complexities of digital product development.

We've explored many principles and strategies for building impactful digital products. Perhaps most importantly, we've understood the importance of empathy and the need to keep the user at the center of everything we do. To build a truly market-winning product, you must deeply understand your users' needs, desires, and pain points, and create a solution that addresses those needs in a unique and compelling way.

After gaining a decent understanding of the users and the market, the next step is to deep-dive into the problem to solve. This involves assessing whether the problem is real, constructing a robust business model, evaluating financial viability, and identifying what drives your product's value proposition.

We've also talked about the importance of sufficient planning in product development, and how to leverage data to make informed decisions and guide the journey. We've also discussed the importance of creating a strong product team and cultivating a culture of collaboration, experimentation, and continuous improvement. You must be willing to adapt and evolve your strategy over time, based on changing market conditions and user feedback. Even after your product's core is built and validated, you must continue to optimize by moving out from opinion-driven decision-making to data-driven decision-making.

We also explored how to harvest the returns of the product your team has developed, how to prepare for contingencies, and nurture customers to bring referrals and up-sales. We have also discussed the importance of retiring voluntarily before someone else makes you obsolete.

In conclusion, building market-winning digital products requires a multifaceted holistic approach. As you embark on your own journey to build a market-winning digital product, remember that success doesn't come cheap. However, by following the approach outlined in this book, you can create high-quality digital products that your target audience loves while also being financially successful.

We wish you the best of luck on your journey to build a market-winning digital product. Stay passionate and committed to your vision, and don't be afraid to take bold risks and pursue big ideas. May your product change the world for the better, and may you find fulfillment and success in your efforts.

The Winning Product

Launching a digital product is an intense undertaking. However, it doesn't mean that you must figure everything out on your own. Why not learn from the experiences of products launched in the past? What if you had a map that can show you where you are and guide you on what to do next. This is what this book is all about.

