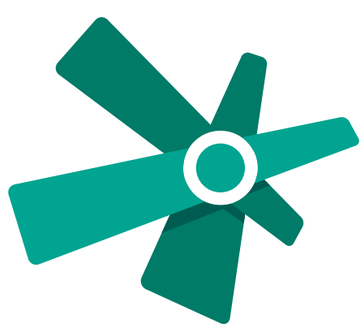
Pivotal



Pivotal **Labs**

Client Product Manager Playbook

v2.0

Contents

Welcome, Product Manager!

Leading a software development team is a multi-faceted and challenging role. We're here to help you and your team build products that deliver meaningful value for your users and your business.

Whether you are a seasoned veteran or a first-time Product Manager working in an enterprise or a startup, this playbook will help you get a sense of the roles, tools, and methods that lay the foundation for a healthy, sustainable, lean and agile product development practice.

By partnering with Pivotal Labs, you are committing to work with collaborators rather than contractors. Together we'll build software based on your continuous input; you maintain full control of product direction, feature prioritization, and release timing.

We will show you how to continuously keep product and business risk low by identifying risky assumptions, and gathering and analyzing qualitative and qualitative data to help inform product decisions.

We will help you balance business goals against user needs and desires and technical feasibility, so that the team regularly ships features that have impact.

We will show you how to set goals and structure regular meetings to ensure fluid communication among developers, designers, stakeholders and sponsors.

We will help you break down big ideas into manageable pieces, establish a reliable way to estimate features, and ensure a predictable cadence of small and frequent releases.

We'll encourage your entire team to provide regular feedback on process and performance, which will drive continuous improvements in efficiency and happiness.

NOTE: If you have questions or feedack about this document, please share with us at client-pm-playbook@pivotal.io

What to Expect

LEARN BY DOING

We teach by showing, rather than telling. You will be a practicing Product Manager, coached by us, from day one. In the beginning, we'll lead, but with the goal of having you lead soon after.

SIT WITH YOUR TEAM

Co-location is crucial to faster feedback loops. By being here, you'll be able to continuously answer questions your team encounter along the way.

CONSTANT COLLABORATION

We believe that the best products are informed by diverse perspectives. You will be partnering with developers, designers, and business stakeholders to inform your product decisions.

YOU OWN THE PRODUCT

It is your responsibility to validate product features and design, collect and prioritize inputs to inform product decisions, write user stories, maintain the backlog, and help run team meetings.

SUSTAINABLE PACE

Healthy product development is a marathon, not a sprint. By maintaining a 40-hour work week, we keep a sustainable and predictable pace that helps us ship value, fast forever.

WE'RE HERE TO HELP

You'll work closely with your Labs PM. When in doubt, lean on us!

Pivotal Labs' Approach to Modern Product Development

Our approach to product development is informed by over 20 years of experience and continuous improvement.

We blend values, principles and practices from three schools of thought:

- Lean Startup
- User-Centered Design
- Agile/Extreme Programming (XP)

Additionally, we apply these philosophies within a structure of "Balanced Team".

Pivotal didn't invent these philosophies. We have many individuals and teams to thank for sharing and refining these ideas. We have developed our own approach, tools and platforms to enable these practices. Taking aspects of each philosophy, we combine them and apply them with discipline, empathy, humility, curiosity and a strong bias to action. We will work closely with you to figure out how to adapt these methodologies for your organization.

Our approach is always evolving as we continue to uncover new and better ways of building great software. What follows is a quick primer to product development which will help you understand how we work.

Lean Startup

Lean Startup is an approach to developing businesses and products in environments with high uncertainty and change. It was created by entrepreneurs Eric Ries and Steve Blank. Ries and Blank based their ideas on lean manufacturing, which was popularized by Toyota as a means of minimizing waste and maximizing productivity.

Most products fail because they are based on flawed assumptions about user needs. Lean Startup emphasizes investigating and validating assumptions about what customers really need, and meeting those needs with as little time, effort and money as possible. Customer feedback is vital during product development. It ensures that we don't build products that customers don't want.

Lean startup also borrows from the scientific method, and assumes we don't know something to be true unless we gather the evidence to validate our belief.

Common practices: *Minimum Viable Product (MVP), assumptions, continuous deployment, actionable metrics, pivot or persevere, build-measure-learn cycles, lean canvas, CI/CD.*

User-Centered Design

User-centered design (UCD) is a product design philosophy that emphasizes designing the product around how the user can, wants or needs to use it, rather than seeking to change the user's behaviors to fit the product.

We consider our user's needs from the very beginning of the product cycle, and validate our assumptions about their behaviors and problems. Using investigative methods like ethnographic study, contextual inquiry and prototype testing we validate and refine our understanding. The whole user experience is considered; we seek to understand the user's full context rather than just the task(s) they would complete using the product.

The design work is done iteratively, and design options are tested and evaluated early and often. Users are involved directly in the design process. Through observation and interviews, we seek to gain empathy for the goals of our users, and use what we learn to ideate and evaluate design options and to inform design and development decisions.

Perhaps the most influential writing done on UCD is Don Norman's "The Design of Everyday Things" and Jeff Gothelf's "Lean UX".

Common practices: *user interviews, personas, scenarios, think-make-check cycles, rapid prototyping, assumptions, validation, Minimum Viable Product (MVP).*

Agile/XP

Agile is a set of values and principles that enable teams to develop software with agility. Agile teams are quick, flexible, and respond to change without sacrificing quality. There are many implementations of this methodology. Most assume collaborative, cross-functional and self-organizing teams that deliver software in an incremental and iterative way.

We do XP which is a specific set of practices that implement agile values and principles. It was created by Kent Beck, who wrote "Extreme Programming (XP)" in 1999.

XP enables teams to write high quality software, ship it often and predictably and be responsive to change. It's a strategy to help teams go fast forever. XP is a system – we only get to experience its full benefits when we implement all of its practices. That's why it's called Extreme Programming!

Common practices: *pair programming, test-driven development (TDD), refactoring, collective code ownership, delaying decisions until the last responsible moment, expecting changes in requirements, frequent communication, small and frequent releases.*

Balanced Team

Our product teams are small, co-located, and multidisciplinary. Teams are organized around goals established by the product sponsor(s) and empowered to define and iterate on a solution that delivers against those goals. The team is empowered to talk to customers, make product decisions, and push code to production. By virtue of sitting next to each other, team members constantly communicate and collaborate with each other. The team is focused on delivering customer value through working software in small iterative releases. It is self-organizing in that it can adapt common tools and practices to what works best for the team. Team communication tends to be informal, favoring spontaneous conversation over lengthy meetings.

We call this type of team "Balanced Team." The concept was created by a group of passionate agile practitioners who sought to blend agile, UX and other product disciplines better. Early proponents of Balanced Team include Jeff Patton, David Hussman, Desiree Sy, Jared Spool, and Lane Halley. Pivots Janice Fraser and Tim McCoy were early participants, and Pivotal Labs hosted one of the first Balanced Team retreats.

Though each project varies, the typical balanced team mix is a Pivotal Labs Product Manager, Product Designer and Engineer matched with full-time client counterparts. When we work within the construct of a balanced team, we ensure that all of these perspectives blend and inform each other so that we build products that are desirable, usable and feasible.

Common practices: *shared vision, physical co-location, pairing across functions, scaling the team up/down based on what the product needs*

NOTE: If there's a term here that is new to you, there's a good chance you'll find definition for it in the gloassary at the end of this playbook.

Balanced Team Roles

Team makeup and size can vary between product teams. What's important is having the right balance of skills and perspectives to make informed product decisions.

Your Role as Product Manager

The PM leads a team to discover and deliver a product that creates meaningful value for their company and users. You'll facilitate decision-making in service of shipping successful features. You'll need to clearly understand who your users are and what they need, what impact the business expects from the product and who your stakeholders are. You'll also need to collaborate closely with your team.

Your Pivotal PM Pair

You will be paired with a Pivotal Labs Product Manager who will guide and support you throughout the project. Your pair will help you understand your role on a Balanced Team and how to take a lean/agile approach to product management. Following the "I do, We do, You do" process over the course of the project, you will gradually take the lead with your pair transitioning to a supporting role.

OTHER ROLES ON YOUR TEAM

These are the most common roles you will see on a Balanced Team. However, your team's roles may differ slightly depending on your product.

EGINEERS

Engineers implement stories from the backlog. They also assist with story estimation in the Iteration Planning Meeting. Engineers guide the implementation and help you understand the technical implications of product decisions. You will help them gain understanding of what product success looks like. You will work together to validate your backlog's prioritization.

ANCHOR

One engineer is designated as the Anchor. The anchor ensures that the team makes sound technical decisions, that engineering practices are healthy and appropriate, that new and junior engineers get up to speed quickly and that the engineering team collaborates well with PM and design. The anchor also ensures that the team collectively owns the codebase and understands the engineering decisions that are made. You will collaborate more closely with your Anchor than with the other engineers on your team.

DESIGNER

Designers deliver value in the form of design decisions. Their job is to deeply understand the users in order to define solutions that are desirable, useful and usable. You will work closely with your designer, pairing on user research, to validate critical user and solution assumptions before adding development work to the backlog.

CLIENT LIAISON

Each team has a Pivotal Labs Client Liaison (CL). The CL is responsible for the client relationship and the overall success of the project from the perspective of both the client and Pivotal. If needed, the CL also helps to unblock the team and provides day to day support.

STAKEHOLDER AND COLLABORATORS

We often work closely with marketing, compliance, risk, infrastructure teams, legal and other stakeholders. A good general rule for groups like this is to assign a point person, establish a regular touch point and provide visibility into each other's roadmaps and/or backlogs with some ability to influence them. If it makes sense, we make members

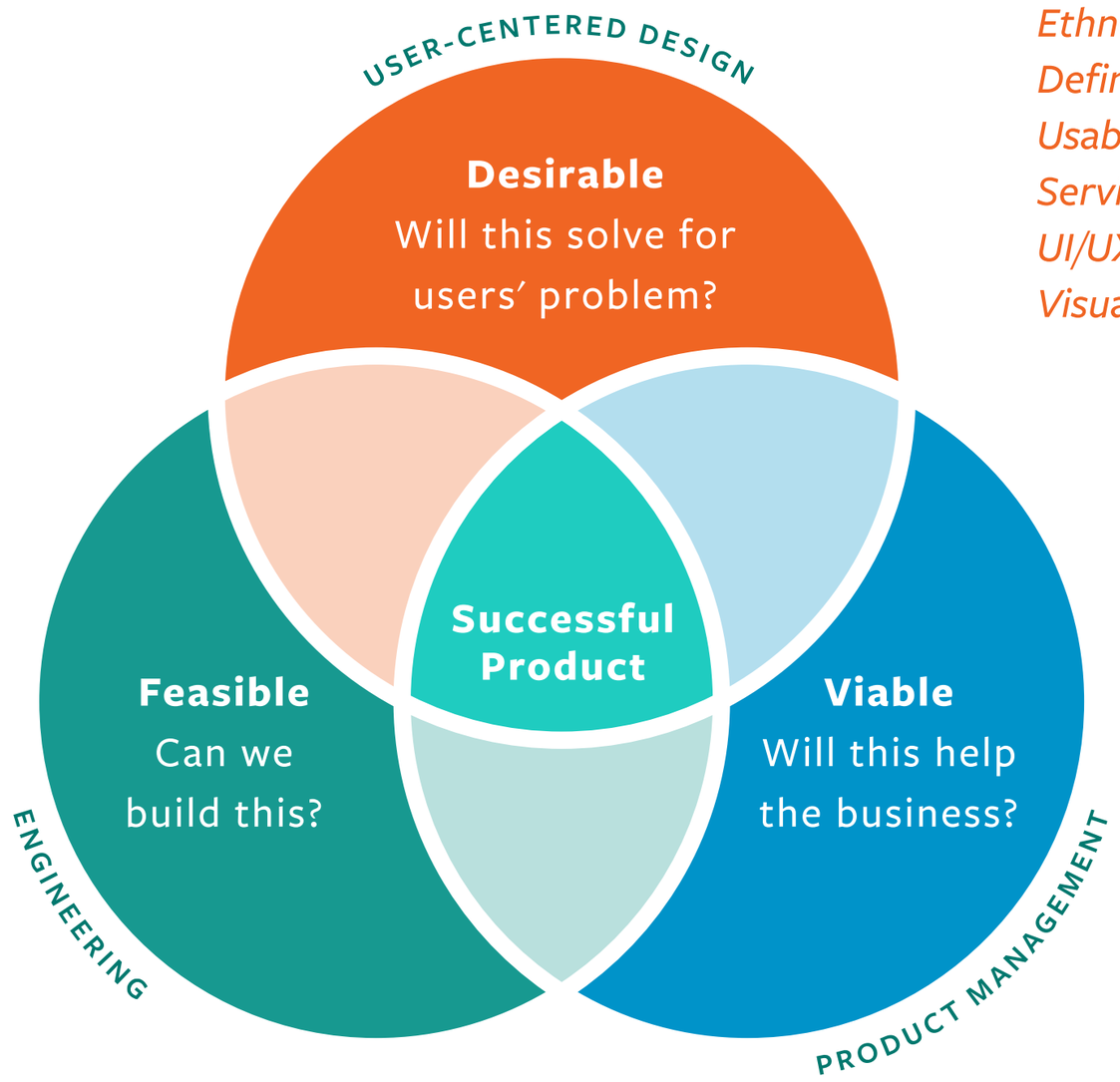
Balanced Team in Practice

To build a successful product, we need to answer three questions: Will it solve for users' problem? Will it help our business? Can we build it well? Lean startup, user-centered design and agile/XP help us identify the best answers to these questions. When we work within the c of a balanced team, we ensure that all of these perspectives blend and inform each other.

User-centered design

Activities:

User interviews
Ethnography
Define personas
Usability testing
Service design
UI/UX
Visual design



Agile/XP

Activities:

Test-driven development
Pair programming
Evolutionary design
Collective code ownership
Retros
Short iterations
CI/CD

Lean Startup

Activities:

Define product vision, strategy and roadmap
Define business model
Define Minimum Viable Product
Identify and test assumptions
Release real product often
Understand customers
Adjust direction based on data
Constrain resources and time

Desirability: Will It Solve Users' Problems?

The product should be something that users want and that solves real problems. Designers' primary question is, "How is the user affected?" Designers more than anyone else on the team will help us solve for "Is this a problem for users?" and "Does this solution solve for users' problem?"

If designers become too focused on user needs and cannot connect with business needs and technical feasibility, they will focus on solution ideas that cannot be implemented and/or will deliver no return on investment.

Viability: Will It Help Our Business?

The product has to support a sustainable business model. The PMs' primary question is, "By solving these specific user problems with these specific solutions, are we creating valuable user and business outcomes?" If product managers become too focused on business and cannot connect with users, they will likely focus on solutions that don't solve any real needs and thus don't get used.

Feasibility: Can We Build It Well?

The product implementation has to be feasible and robust. Engineers' primary question is, "What technical implementation will satisfy the project and product goals best?" Engineers help us debate potential solutions and the technology constraints and realities of what we're trying to do.

It doesn't help for engineers to come up with feasible solutions that don't solve problems for users. That is not developing good software. And if we hone in on a solution that is desirable and viable and yet not feasible, we've failed.

Team Rhythm

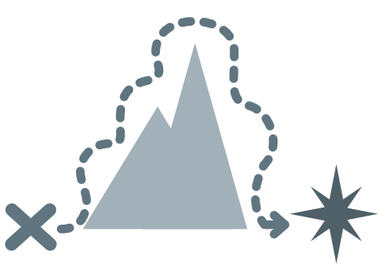
A healthy lean and agile team has a strong and consistent rhythm. Each week is punctuated by a small set of standing meetings.

Daily Meetings	Meeting Purpose	Your Role
Project Standup	To check in on everyone's progress, plan, and blockers.	• Understand blockers and their implications on the current priorities • Remind everyone of action items and upcoming milestones • Make sure team members take turns faciliating standup
Office Standup	To kick off the day with new faces, helps, interestings, and events.	• Ask for help if you need it • Announce new teammates or guests • Share things you've learned that could help other teams
Weekly Meetings	Meeting Purpose	Your Role
Iteration Planning Meeting (IPM)	To estimate complexity of the week's backlog of prioritized user stories that the developers can pick up for implementation. (PM leads.)	• Communicate the user value and business value of each story • Clarify any confusion and update stories • Confirm the priority of stories with input from the developers
Optional: Pre-IPM	To ensure the week's backlog includes stories that are ready to be estimated and discussed in IPM.	• Ensure stories are well-written • Confirm the splitting of stories with input from the developers
Sponsor and Stakeholder Update	To share the team's progress in terms of validated learning and working software, demo the working product, raise blockers, give an update against KPIs and share what the team plans next.	• Prepare the agenda with your anchor and designer • Use your product roadmap to explain what the team is working on and why • Frame your demo in the context of the outcomes defined in your product roadmap • Speak to the empirical data guiding the team's decisions • Speak to your assumptions: those you have tested and those you plan to test soon • If blocked, explain what the team needs, why and by when
Retro (short for "Retrospective)	To create a safe space for the team to celebrate the past week's successes, discuss points of confusion, and reflect on challenges	• Openly and honestly share your experiences from the week • Dig deeper on items raised by others to understand root causes

Your Role as Product Manager

Product Managers, in close collaboration with their teams, help their company ship the right product. By the time you go back home, you'll be familiar with a set of tools and techniques that will help you build products by defining and iterating towards the right outcomes and continuously validating and iterating towards the right solution in a way that will enable you to be responsive to changing user demands and market realities.

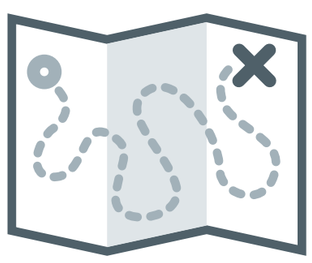
Articulate the Product Vision and Product Strategy



So that we know why we're undertaking the effort of building the product, and what our plan for doing so is. **PAGES 10 – 12**

Establish an Outcome-Oriented Product Roadmap

So that we have clear outcomes to prioritize our work against, and can communicate to sponsors and stakeholders how the product is likely to evolve over time. **PAGES 13 - 15**

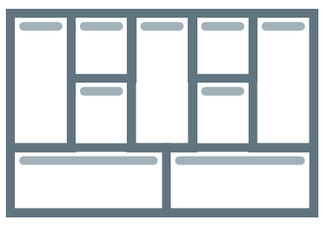


Establish and Track Against Measurable Objectives

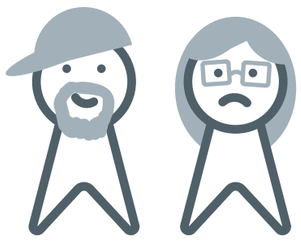
So that we know whether we are on the right path or not.
PAGE 16

Continually De-Risk Product Direction

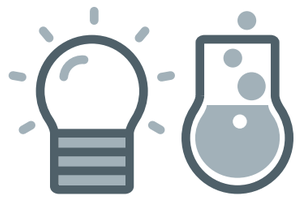
So that we only build the features we should build, based on validated customer problems and business opport-unities, and increase the likelihood that the product will be successful in the market. **PAGES 17 - 24**



Lean Canvas



Personas



Assumptions & Experimements

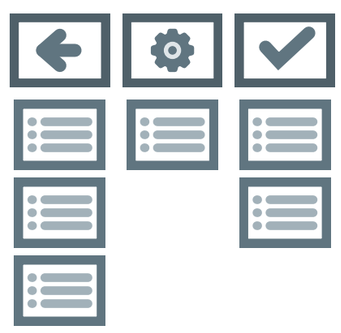
Minimum Viable Product

Prioritize Features

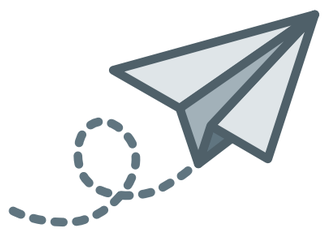
So that we are always working on the most valuable thing.
PAGES 25 - 26

Manage the Backlog

So that the product team has a shared focus across tactical design and development work.
PAGES 27 - 35



Decide When to Ship Software



So that our customers can experience the benefits of our product as soon as possible, and we continue to gather valuable feedback about how to iterate on the product.
PAGE 36

Help the Team Maintain Sustainable Pace

So that we can go fast forever and not burn out or lose motivation. **PAGE 37**

Communicate Effectively

So that product team, stakeholders and sponsors are all aligned and we maintain momentum.
PAGE 38

Pivotal **Labs**

Understand the Product Vision

The product vision describes the future we are trying to create through our product, and how our product intends to contribute to our company's larger purpose.

Vision

The product vision describes the **WHAT** and the **WHY**. It's highly aspirational and long term, and as such realizing it may take several years (~5 years or more).

EXAMPLES

Amazon: *Our vision is to be earth's most customer centric company; to build a place where people can come to find and discover anything they might want to buy online.*

LinkedIn: *To connect the world's professionals to make them more productive and successful.*

Spotify: *To enable people to have music moments everywhere.*

The Product Vision is Your True North

The product vision states what the product could ultimately become in support of your company's overarching purpose. It reflects a team's or company's core values, motivations and intentions. It's often referred to as a team's "true north", defining the product's direction and guiding the team's every decision and action.

Your product vision should communicate:

- Who you're creating the product for
- What needs or desires the product will address
- What benefits the product will create

Why You Need a Product Vision

Ideation and experimentation alone will not enable us to create great products. Without a vision, we're left to aimlessly iterate towards some unknown destination. If we don't know where we want to go, how would we know if we're getting there?

Your product vision helps you tell the story of your product. It explains what you hope to achieve and sets the direction for where you are going.

A compelling product vision will:

- Help others understand your product and why it matters
- Align, guide and motivate everyone involved to make the product successful
- Create focus and enable effective collaboration in the midst of rapidly changing conditions

Who Defines the Product Vision?

Depending on how your organization is structured, your product vision might be defined by senior product or business leaders like CEOs or CPOs. At a startup, it might be the founder. At an enterprise it might be a PO, Director, or VP. If this is the case, you need to make sure you clearly understand their vision. Write down what you understand it to be and validate it with them.

Clearly Display Your Product Vision

Post your product vision somewhere in your workspace where it's clearly visible to everyone! This way, it'll be a constant reminder of what you're working towards.

Craft a Compelling Product Vision

Getting to the right vision statement for your product can be challenging and take some time.

Best practices for creating a good product vision

Keep it short and succinct. It's important that your product vision is easy to understand, remember and communicate. Try to keep it to one or two sentences.

Align it with your company vision. Your vision should clearly communicate how your product will contribute to your company's larger purpose. If your company has one product, then your company vision also makes for a great product vision. If your company has a portfolio of many products, then you should align your product vision with the reason for being of your particular business unit or group.

Focus it on the user. The product vision should answer who the product is intended for, what needs or desires the product satisfies for its users, and what benefit(s) the user can expect to experience by using the product.

Stay away from specific solutions. Your vision should describe the positive impact your product is intended to have. Your vision shouldn't explain what your product is or how it works – you will figure these things out through your product strategy and product roadmap. If your vision makes any statements about particular features or technology choices, it'll limit you later and prevent you from being flexible to new information and changing circumstances.

Make it big and ambitious. A bold product vision will help you remain flexible in what product strategy you pursue and how. It will enable you to expand and evolve both your product and your business over time as the market and what customers need and want changes.

Make sure it's inspirational. The product vision should be something that people care about and can connect with. If the product vision clearly states the benefits you're looking to create for others, you're already halfway there. A vision that matters will motivate the team when things get tough, make people excited to work on the product, help you attract new team members and connect with the right customers.

Make it unique. Explain what sets your product apart from other alternatives, and why it matters. This too will help you connect with the right customers, as well as attract new team members who want to contribute to making the product vision real.

Iterate it. Make sure that even existing products, not just new products, have a clear vision statement. If your vision is no longer aspirational, motivational or rings true to customers, articulate a new and better vision statement.

Getting to a good product vision may take time – validation helps

It may feel challenging to come up with a good product vision. The more time you spend talking to your customers and users, as well as the people running your group and/or company, the clearer your vision will become. Use their feedback to understand whether your vision resonates, and iterate until you get to something that feels right.

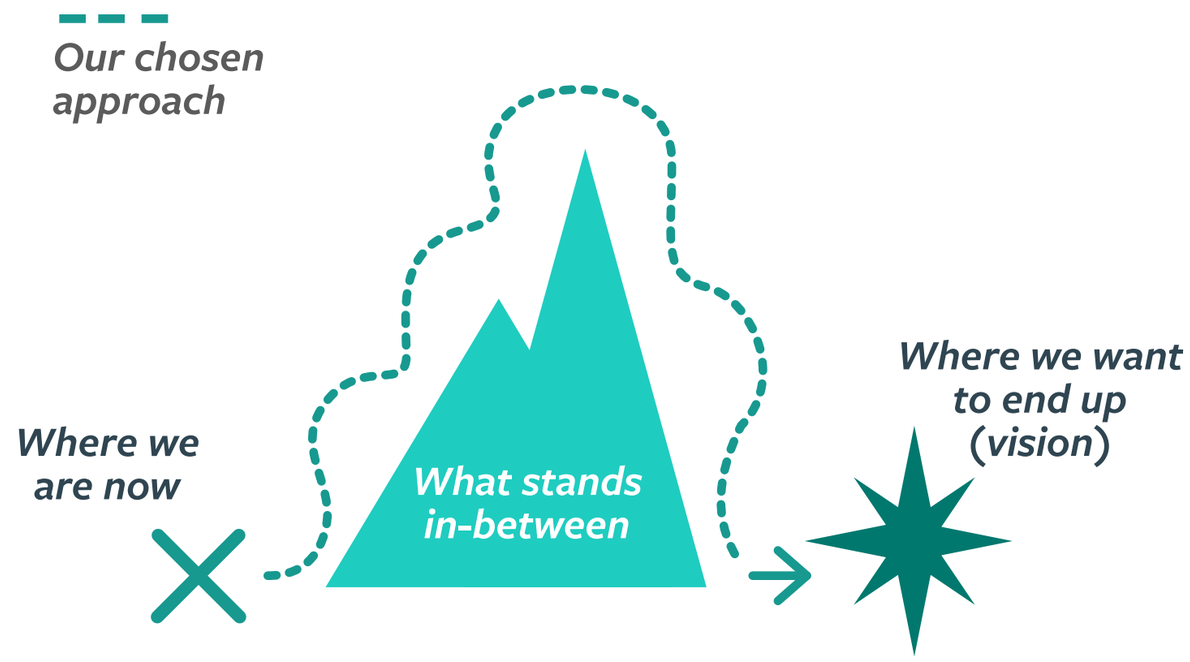
☐ **ACTION:** Work with your Labs PM pair to capture your product vision statement. Put it up where the team sits so that everyone can see it.

Understand the Product Strategy

Your product strategy is a high level plan for realizing your product vision. It defines your path forward by addressing the challenges you must overcome and the initiatives you believe will help you do so.

Strategy

The product strategy explains **HOW** we will realize our product vision. It might span several years.



Strategy is the Practice of Figuring Out How to Get From Here to There

The product strategy is our sequence of products we plan to deliver on the path to realizing the vision. – Marty Cagan

Product strategy is about figuring out what product to make such that we achieve one or more goals under conditions of high uncertainty. Consequently, strategy is also about knowing what not to build, what to say no to and why.

Whether you're part of a large enterprise or a startup, your product strategy should always be in service of your product vision which provides the context for why you're building the product in the first place. The vision is what inspires us and the strategy is what gets us to the vision.

The product strategy covers a set of choices you make about the path forward:

- Who your key customers and users are
- Which problems you intend to solve for your key customers and users in order to realize your business objectives
- How you are different from other solutions
- Which market(s) you're focusing on, vertical as well as geographical
- How you price and market your product
- How you will measure success

Your strategy will, and should, change as you start to implement it. You will begin to see whether it's effective or not, and you will also learn more about your market – and as you uncover new information and/or achieve your goals, you'll want to change or evolve your strategy.

Strategy is About Making Choices About Where We Should Invest

As PMs, we're inundated with request and ideas. How do we know what to say no to? We can't do everything, because the time, money and resources we have available to invest in our product are finite. Additionally, a product that does everything for everyone will do nothing well for anyone – and by the time we launch it, our competition may have raised the stakes beyond our grasp.

We need some way of knowing that we're working on the right things and that we're making good progress against our vision. A product strategy helps us identify the right objectives and decide where to invest to best achieve those objectives.

Who Defines the Product Strategy?

It might be you, or it might be your management team – your group/business unit director, or your VP of product.

☐ **ACTION:** Work with your Labs PM pair to articulate your product strategy. Put it up where the team sits so that everyone can see it.

Plan Around Outcomes, Not Features

Your roadmap communicates how you intend to realize your strategy. It aligns the product work to overarching business goals and helps communicate the current and future state of your product.

Roadmaps Facilitate Alignment and Cross-Team Coordination

Your roadmap might be your most important communication tool as a PM. It creates visibility into your work for people outside of your immediate product team. It helps product sponsors, stakeholders like sales, marketing and customer support as well as other product teams understand your product vision, your goals and the steps you intend to take to achieve those goals.

You can and should use your roadmap to help stakeholders, collaborators and product sponsors understand what is coming up next and to coordinate joint efforts and dependencies with other teams. By clearly communicating your goals and priorities, you will help surface important conversations early and keep the product work aligned with the business' needs and priorities.

Be Stubborn on Vision and Flexible on Details

If you've seen a product roadmap before, chances are you've seen a comprehensive plan spanning several years, with detailed feature specs, timelines, work estimates and ROI predictions.

Lean and agile product teams approach product planning differently. No plan survives its first encounter with reality. We learn more from building and releasing features than from planning. Additionally, we reduce risk in our deployments by deploying incrementally with smaller net changes. Remember, we work in build-measure-learn cycles! Therefore, we should spread our planning over the course of the project instead of front loading our work with detailed plans.

What matters is that we get to the right place in the end, not that we follow the exact path we envisioned when we set out on our journey. Rather than try to optimize for predictability in an inherently unpredictable world, we optimize for learning and responding to change as quickly and cheaply as possible. With the product vision as our north star, we articulate a high-level direction for getting to that vision (our product strategy) and define a plan for our immediate next steps (our product roadmap). As we move forward and learn more than what we knew when we started, we refine and adjust our roadmap accordingly.

Outcomes Matter More Than Features

The roadmap should emphasize the results we want to create – like improved product and business metrics – rather than the features we believe should be part of our product. This approach creates agreement on what success looks like while enabling us to be flexible in our solution. Being outcome-oriented enables us to iterate towards the best solution as we learn more about the market, our customers and users and what's technically feasible. In addition to focusing us on the impact we want to have through our product, an outcome based approach to planning helps us:

- Create shared understanding around why we are building the product
- Create alignment between stakeholders, sponsors and the product team
- Prioritize feature ideas and requests better, because our team mandate is clearer
- Feel more engaged in our work by being empowered to solve a problem
- Figure out which parts of our strategy are working and which are not

Create an Outcome-Oriented Product Roadmap

Your product roadmap describes how your product is likely to evolve over time.

Work backwards from your vision and strategy

Your roadmap should consist of sequential goals – your strategy – that help build towards your vision. To define your goals, identify the key challenges you need to overcome first. Also look at any known strategic outcomes that have been defined by the business. Outcomes are the important measurable impacts we want to create for our customers and our business, like increased market share, more paying customers or reduced tech debt.

You should have clarity on:

- The problems that need to be solved
- How you intend to solve them
- How you will know you've solved them

Identity the work that needs to get done first

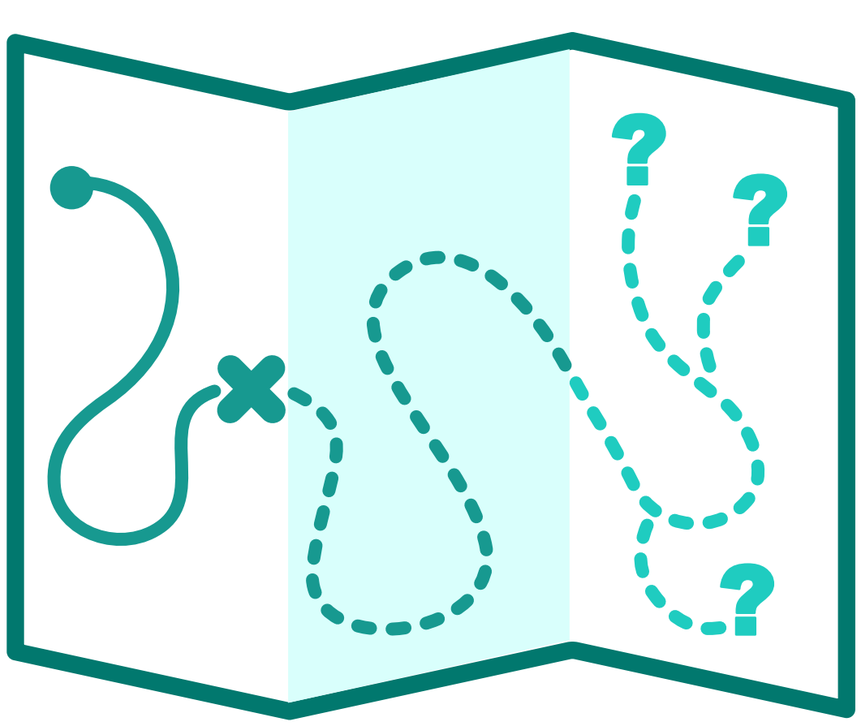
Once you know your business goals, capture your assumptions about how you might reach those goals. You probably want to identify and validate your assumptions about what customer problems you need to solve, the features that solve for those problems, how you might design and implement those features, the business value the solution would deliver, and possibly also how you'd market and operationalize the product.

When you prioritize assumptions to validate and product work to execute on, think about:

- What dependencies do you need to solve for first?
- What do you need to learn first?

This will help you create a list of learning objectives, features and activities you believe will help make an impact against your business' strategic objectives.

☐ **ACTION:** Make time with your Labs PM pair to create an outcome-oriented product roadmap. Make sure you include input from design and engineering.



Plan across three horizons

To communicate the rough order of your priorities, organize your roadmap in three time horizons:

Current. This is work you are doing now.
Near Term. This is work coming up soon.
Future. This is work you're thinking of doing but need to research more.

It's most important to have clarity on your first few goals. Your roadmap will evolve and change over time, and the farther out a goal is, the more about it is likely to change before you get to it.

Roadmap Design Principles

1. Communicate the direction you plan to go in to realize your vision.
2. Assume uncertainty and change, optimize for learning and responding.
3. Frame the work around intended or desired outcomes, not outputs.
4. Make sure your roadmap is easy to update and share.

Example: HomeWiFi's Outcome-Oriented Product Roadmap

Meet HomeWiFi, a provider of Home Wi-Fi systems.

Company Vision

HomeWiFi's vision is to provide a superior in-home wifi service that just works.

Current State

HomeWiFi targets affluent homeowners who own one or several large properties. Through customer research, HomeWiFi has learned that these homeowners want home connectivity with a high level of convenience, reliability and speed as they use their networks primarly for home security and secondarily for communication and entertainment.

HomeWiFi gets a sizeable share of their new customers through the third party installer market. These installers help customers get their Wi-Fi network up and running. They also monitor it post installation on behalf of the customer to ensure that everything is running smoothly. Homeowners often have a long-standing relationship with their installers. They expect a responsive high-touch customer service experience provided directly by their installer, either in person or remotely.

The Opportunity

HomeWiFi currently doesn't offer third party installers any tools for working with its systems, making its systems more challenging to install and support and compromising upper-end installers' promise to their affluent homeowner customer segment to provide superior customer service.

To be more competitive in the upper-end installer market, HomeWiFi has decided to develop an application to help installers troubleshoot networking issues.

Product Vision

Become the tool that enables HomeWiFi installers to be the smartest and most efficient installers in the industry.

Product Strategy

Create a dashboard that enables installers to detect, troubleshoot and resolve networking issues remotely before the customer notices.

EXAMPLE: HOMEWIFI'S OUTCOME-ORIENTED PRODUCT ROADMAP

	CURRENT	NEAR TERM	FUTURE
DESIRED OUTCOMES	Installers can solve customer issues without making customer visits	Installers do not need to call HomeWifi tech support	Installers can resolve customer issues in under 60 min
KEY CHALLENGES TO OVERCOME	• Installers have no visibility into customers' network, node and device connectivity status • Installers cannot reboot customer networks remotely	• Installers don't have a way to push firmware upgrades to nodes • Installers forget to complete their registration process	Installers don't have visibility into historical information about the network
SUCCESS METRICS	80% of issues are resolved remotely	80% decresase in tech support calls from installers	90% of issues are solved in 60 minutes or less
FEATURES	(*To be defined, designed and validated*)	(*To be defined, designed and validated*)	(*To be defined, designed and validated*)

Establish and Track Against Measurable Objectives

To know whether we are on the right track to deliver on our product goals, we need to establish clear objectives and metrics.

Goals Change Through the Product Life Cycle

Goals have a long timeframe, articulate a high level outcome we strive towards, and may not be directly tangible or measurable. They are your desired outcomes and often strategic in nature, like increase sales or keep customers happy. Your goals will likely depend on what stage you are at in your project and with your product, which in turn will determine what question(s) you're looking to answer. Early on in the product life cycle, we seek to validate a product. Later on when we have a validated product, we look to optimize our business model as much as possible.

Objectives Make Goals More Achievable

Objectives are the specific, measurable, achievable and tangible actions we take to realize our goals. One goal can be broken down into several objectives. This helps us break down a big challenge into many smaller challenges. A good practice is to make sure that your objectives are SMART: specific, measurable, attainable, realistic and time-bound.

Clear Goals and Objectives Help Minimize Waste and Maximize Value

It's important that we have complete clarity on why we invest time in various activities, such as talking to customers, exploring technology X or building feature Y. Everything we spend time, money and effort on should intend to help us realize an important objective.

Metrics Enable us to Track Progress

Metrics are numbers that define some standard measurements that are important to us. Metrics that help us assess how well we're doing against key business objectives are called Key Performance Indicators (KPIs). It's important to take note of your baseline metrics before making any changes, so that you can compare the before and after or run split tests.

Beware of Vanity Metrics

Good metrics are actionable:

- They tie back to specific and repeatable tasks that you can improve upon, and to your product and business goals.
- They are expressed as a ratio of two things, like % of users who make a purchase.
- They have a time component to enable you to see change over time so that you know whether you need to take action.

Vanity metrics are numbers that tell us about the current state and often make us feel great because they are big, but offer no insight into how we got to where we are or what to do next. Examples: total # of visitors, total # of downloads.

Tying it All Together

Example: Online Retailer

Goal: Improve customers' satisfaction with our ite experience

Objective: Improve site performance

KPI: Increse site availability measured over the last month from 97.5% to 99.5%

Example: Online Media Property

Goal: Improve profitability

Objective: Increase banner ad revenue

KPI 1: Visitor loyalty (average # returning visits/ month)

KPI 2: Click through (average # banner ad clicks, per user per month)

Example: On-Demand Car Service

Goal: Create a mobile app customers love

Objective: Become a top rated (>4.5 stars) app in the iOS market place

KPI: Averge iOS app store rating

DE-RISK

Continually De-Risk Product Direction

We'll have lots of ideas for our product. How do we know which ones are good, and which ones are bad?

Untested Assumptions Create Risk

Before executing on product ideas, teams should clarify their thought process and test their ideas so that they move forward with confidence. We need to answer some important questions before we are ready to build a product.

Types of Questions

Product risk

- Do we have a meaningful problem to solve?
- Can we validate a minimum viable solution that people find useful?
- Can we scale that solution?

Customer Risk

- Are there enough people who have the pain?
- Can we effectively reach these people?
- Can we effectively reach these people at scale?

Business Risk

- Will a sufficient number of people pay enough for the product?
- Can we define cost and revenue structures that enable us to have a sustainable business?

Technology Risk

- Can we implement the solution well?
- Can we maintain and continue evolving the product over time?

Team Risk

- Can we take action and move forward effectively? Do we have access to users? Do we have access to the business context we need? Can we ship to production? Can we maintain a sustainable pace as a team?
- Do we have a clear shared focus, so that we can move quickly and do the right things?
- Do we have the right skills and perspectives available and access to all the inputs we need to make good decisions?

Our initial attempts to answer these questions are assumptions until proven valid through empirical data. It's when we make product decisions based on assumptions, rather than on careful analysis of data, that we accumulate risk.

How Do We De-risk Our Product?

Generally speaking, we want to:

- **Validate a meaningful problem.** This is a need or desire that is strongly felt by a sufficient number of people.
- **Validate a solution that is feasible, desirable and viable.** This is a way to address the need or desire such that people are satisfied, and will pay for it.
- **Validate a viable business model.** This is about defining a business around our product.

This requires us to continuously prioritize our riskiest assumptions, design and run experiments, analyze the experiment data and determine whether our assumptions were validated or invalidated.

Create a Lean Canvas

The lean canvas helps us stay disciplined about documenting and validating our riskiest business model hypotheses.

The Product is Only One Component of a Successful Business

It's important to remember that your product alone doesn't make for a successful business. You need also need to make sure that the product is driving business impact. The canvas helps us capture and systematically de-risk the riskiest aspects of creating a business. Think of the lean canvas as a roadmap for validating and developing a business plan. It's an important tool to help us stay aligned on what a successful business looks like.

The lean canvas was developed to help entrepreneurs define new products. It can also be used successfully to summarize how an existing product drives business impact, and where there may be opportunities for optimization.

Components of a Lean Canvas

- The **problem** is our customers' top 3 underserved or unaddressed needs.
- The **solution** is the top 3 features we believe will address our customers' problem.
- **Key metrics** are the key activities we measure.
- The **unique value proposition** is a clear and concise message that states how we are unique and worth buying/engaging with.
- **Unfair advantage** is something we have that can't be easily bought or copied.
- **Channels** is our path to our customers.
- **Customer segments** are our target customers.
- **Cost structure** and **revenue streams** is how we plan to spend and make money.

☐ **ACTION:** Make time with your Labs PM pair to create a lean canvas for your product. Make sure you include input from design and engineering.

<table>
<tr><td rowspan="2">PROBLEM

EXISTING ALTERNATIVES</td><td>SOLUTION</td><td rowspan="2">UNIQUE VALUE PROPOSITION

HIGH-LEVEL CONCEPT</td><td>UNFAIR ADVANTAGE</td><td rowspan="2">CUSTOMER SEGMENTS

EARLY ADOPTERS</td></tr>
<tr><td></td><td></td></tr>
<tr><td colspan="2">COST STRUCTURE</td><td colspan="3">REVENUE STREAMS</td></tr>
</table>

DE-RISK

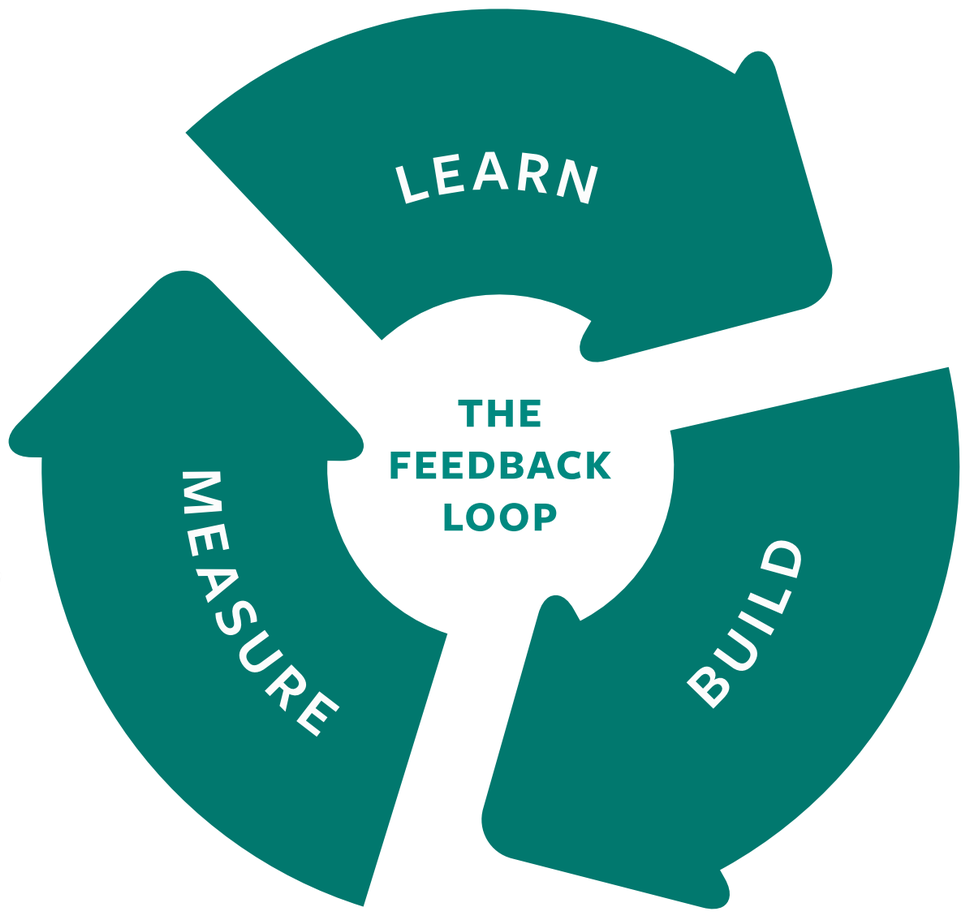
The Product Development Cycle

We use a ″Build-Measure-Learn” feedback cycle to continuously turn uncertainties and assumptions into facts. Starting with ideas, we identify our riskiest assumptions, build simple tests, run those tests, analyze the test data and then use what we learn to inform our next steps. This means that we don’t treat product definition, design, development, and testing as separate phases. Instead, we do all of these activities in parallel via short and frequent cycles throughout the product life cycle.

Learn

From data analysis whether to pivot or preserve

Activities: *Validate/invalidate riskiest assumptions about personas, top problems to solve, existing solutions, desirability of solution, feasibility of solution, viability of product, business model sustainability, business model optimizations.*



Measure

Results of tests

Activities: *Customer observation and interviews, split tests, real-time monitoring, funnel analysis, cohort analysis, search engine marketing.*

Build

Small and simple tests to collect metrics

Tests: *Interview script, value proposition, prototypes, Minimum Viable Product, usabiliy tests.*

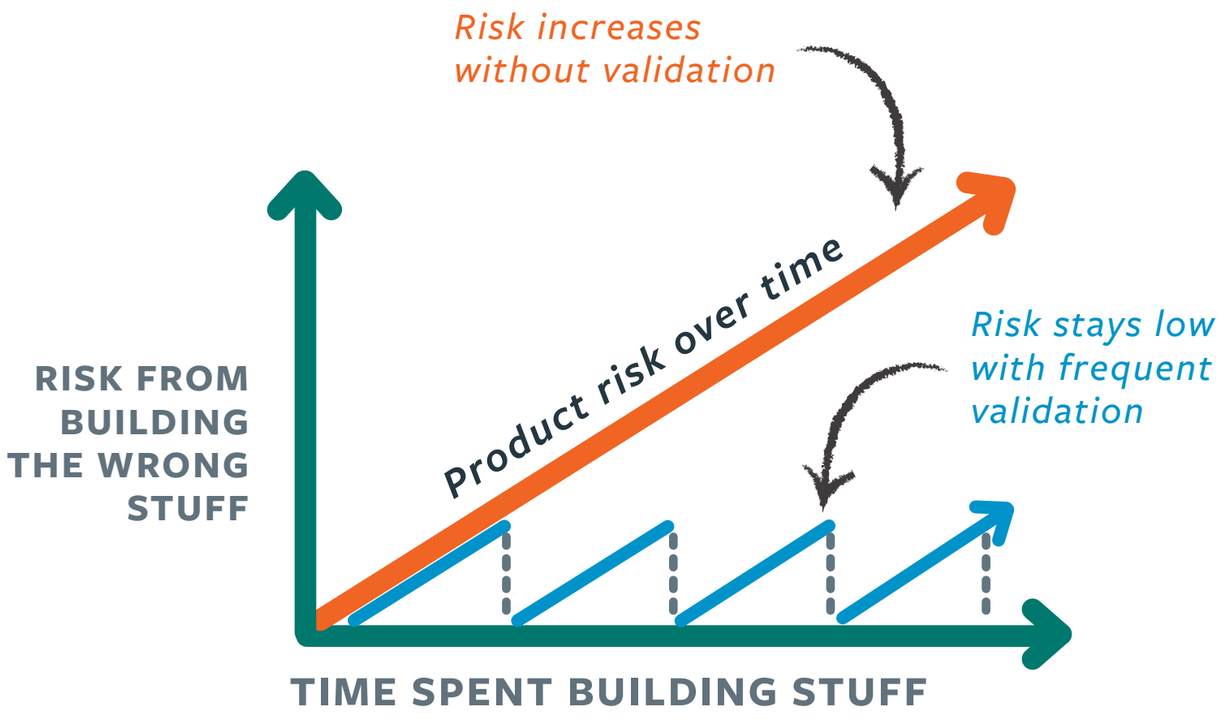
Why?

We work within this cycle to reduce risk of spending time, money and effort building software that delivers no meaningful or impactful value to the business or the user.

Measuring Progress

We look at two things to determine if we are making progress:

- **Validated learning,** which de-risks our product and business
- **Working software,** which delivers value to our customers, and thus to our business



Pivotal **Labs**

DE-RISK

Test Your Leap of Faith Assumptions

It's no longer good enough for a Product Manager to say, "I think users want this feature." Instead, you need to ask, "What outcome do we predict this feature will have?" and validate your answer with empirical data.

Hypothesis-Driven Product Management

Hypothesis-driven product management is the practice of treating the development of new products as running a series of experiments. Instead of formulating requirements, we formulate hypotheses along with some validation criteria that state how strong of a signal we need to consider the hypothesis true. We use what we learn from each experiment to iterate on our ideas until we get where we want to be, or, until we determine that the product isn't viable and cancel the effort.

Leap of Faith Assumptions

These are the riskiest assumptions we make about our idea; if we get them wrong, our product will fail. Our riskiest assumptions should align with the business outcomes or product goals we are working towards. By identifying our riskiest assumptions early and testing them through experiments, we dramatically reduce the uncertainty associated with our product ideas. Instead of trying to get it right from the start, we try to get it right one small piece at a time. We test our risky assumptions systemically in order of potential impact:

Problem/solution fit: Have we found a problem worth solving? Can we solve it?
Product/market fit: Have we built something the market wants?
Scale: Have we found a sustainable business model?

Experiments Test Our Assumptions

A lean experiment is the smallest experiment we can run to quickly test our assumption. We start small and fast, and then increase the scale and scope of our experiments over time.

☐ **ACTION:** Discuss different experiment types with your Labs PM pair.

An experiment consist of three parts: a hypothesis, a test and validation criteria.

The **hypothesis** is falsifiable version of our assumption. Remember to make sure you are only testing one variable in each hypothesis, otherwise you won't get reliable data.

The **test** is how we intend to test our hypothesis, and provide it true or false.

The **validation criteria** is the signal we need to see to consider the hypothesis true.

Experiment Template

HYPOTHESIS
We believe that <this capability> will result in <this outcome>.

TEST
We will do/make <this test>.

VALIDATION CRITERIA
We will know that our hypothesis is valid if we observe/measure <this outcome>.

Validation Criteria

You need to determine how much evidence is enough for you to consider an assumption validated. These are some general rules to consider:

- During discovery, talk to as many people as possible
- During usability testing, 5 people will help you identify 80% of the issues
- If your organization is new to hypothesis-driven product development, define your validation criteria based on what you need to feel confident to justify the product decision

DE-RISK

Example: A Lean Experiment For HomeWiFi

HomeWiFi has identified an initial set of Leap of Faith assumptions. They decide to test the following assumption first because they believe is poses the greatest risk to the success of their installer dashboard, if they get it wrong.

Leap of Faith Assumption:

Customers feel comfortable sharing ownership of their networks with their installer.

Installers currently use a customer facing app to install a customer's new Wi-Fi network. The installer will typically make themself the owner of the network and maintain that ownership post installation. This ensures the installer continued network access so that they can monitor and troubleshoot it through the consumer app.

Because the installer accesses private network data without receiving explicit permission from the customer, this becomes a big privacy issue.

In order to give installers what they need to provide the best possible service – reliable access to the customer's network – and in order to give the customer what they need to feel safe regarding who has access to their data – the ability to grant and revoke network access – HomeWiFi must validate their assumption that customers are willing to share network ownership with their installer.

HomeWiFi plans to test this assumption twice before building the new installer dashboard app. First, they test it without building any software at all (Lean Experiment 1). Assuming the first experiment passes its validation criteria, HomeWiFi will run a second experiment in which they build a new feature in the consumer app (Lean Experiment 2). These experiments enable HomeWiFi to test a core assumption relatively quickly and cheaply.

LEAN EXPERIMENT 1

HYPOTHESIS

We believe that giving home owners the ability to share network access
Will result in an increase of network ownership transfers from home owner to installer

TEST

We will continue to use the consumer app to install new customers' networks, and while doing so, ask the customer if they agree to transfer network ownership to the installer (without building any new features for this in the app).

VALIDATION CRITERIA

We will know that our hypothesis is valid if we observe 90% of customers agree to transfer network ownership to the installer, during network installation

LEAN EXPERIMENT 2

HYPOTHESIS

We believe that giving home owners the ability to share and revoke network access
Will result in an increase of network ownership transfers from home owner to installer

TEST

We will enable customers to receive and approve/reject ownership transfer requests from their installer via the customer app

VALIDATION CRITERIA

We will know that our hypothesis is valid if we

- See 90% of customers transfer their network ownership to their installer during network installation
- See 75% of customers maintain a shared network ownership with their installer one month post installation

DE-RISK

Experiment Techniques

There are many types of experiments we can conduct to test our hypotheses. Which experiment type is right depends on what we're looking to learn.

Customer Interviews

Help us validate assumptions about

- What our customers' problems are
- How they want to solve their problem
- What is standing in their way
- Who our customer segments are

and identify early adopters.

Low Fidelity Prototypes

Rough, early, testable representations of concepts that help us validate those concepts early on in the design process.

High Fidelity Prototypes

Detailed, early, testable representations of concepts that are clickable/tappable that help us validate those concepts midway and late in the design process.

Walking Skeleton Test

A tiny implementation of the system that performs an end-to-end function. Helps us validate technical feasibility.

Concierge Test

A technique to replace a complex automated technical solution with humans who directly interact with the customer. Helps us validate whether anyone wants our product.

Wizard of Oz Test

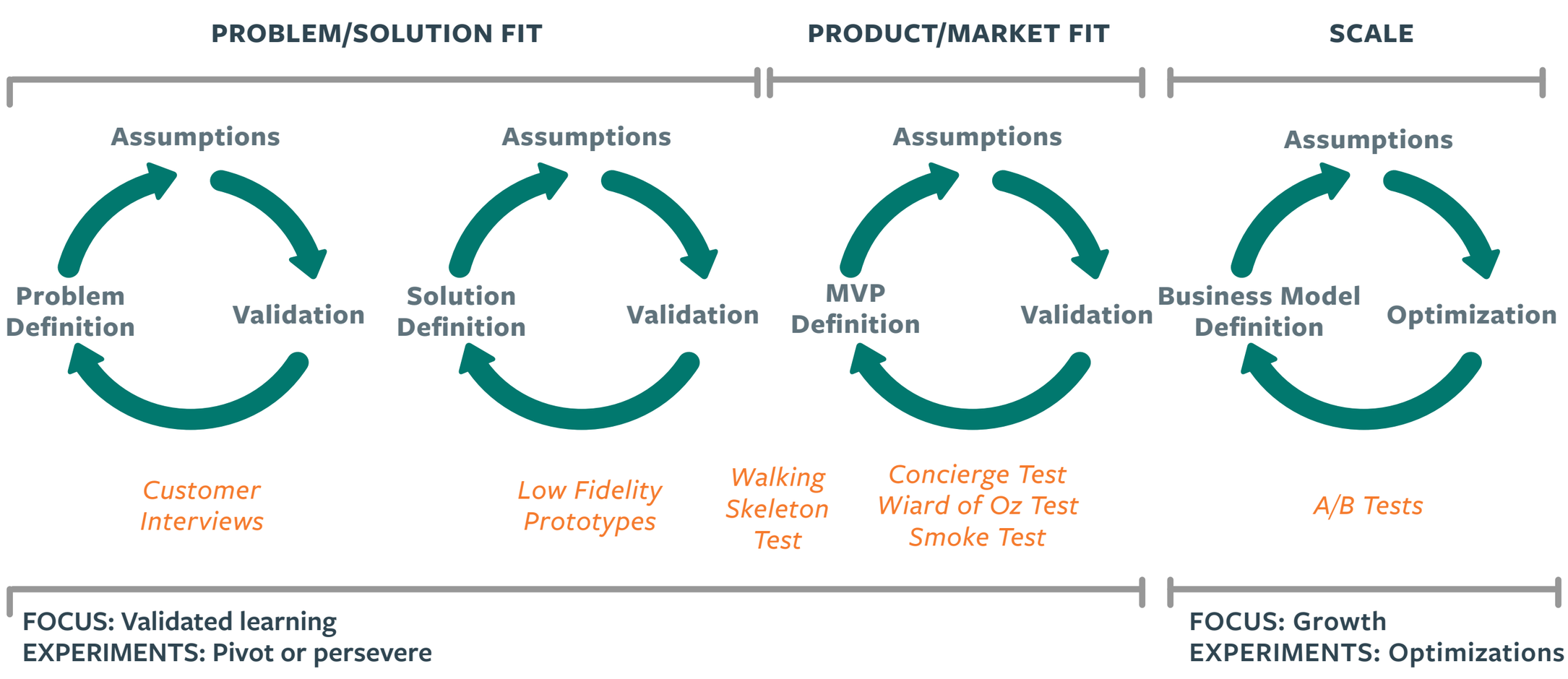
A technique to replace the product backend with humans. The customer believes they are interacting with an automated solution. Helps us validate whether anyone wants our product.

Smoke Test

Commonly a website that describes the product's value proposition and asks customers to sign up for the product before it's available. Helps us validate whether anyone wants our product.

A/B Test

A comparison of two versions of a product or feature to see which one performs best. Works best with large sets of users for small incremental optimizations of an experience and business model.

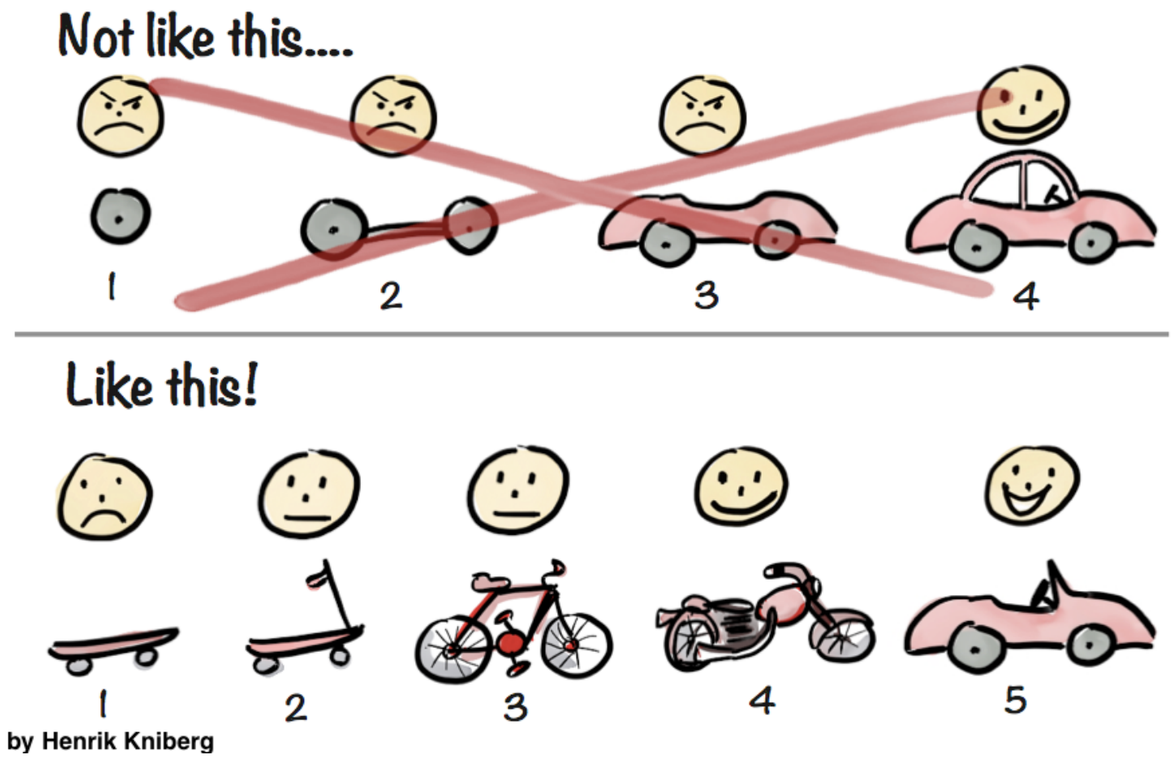


Define Your Minimum Viable Product

The Minimum Viable Product (MVP) helps us determine whether our product should exist.

MVP

The Minimum Viable Product is **THE SMALLEST POSSIBLE VERSION OF OUR PRODUCT** that delivers on the product's value proposition. If it fails, our losses are small. If it succeeds, we'll make it bigger and test it again.



The MVP is a Learning Milestone

The most expensive way to test your idea is to build production quality software – Jeff Patton

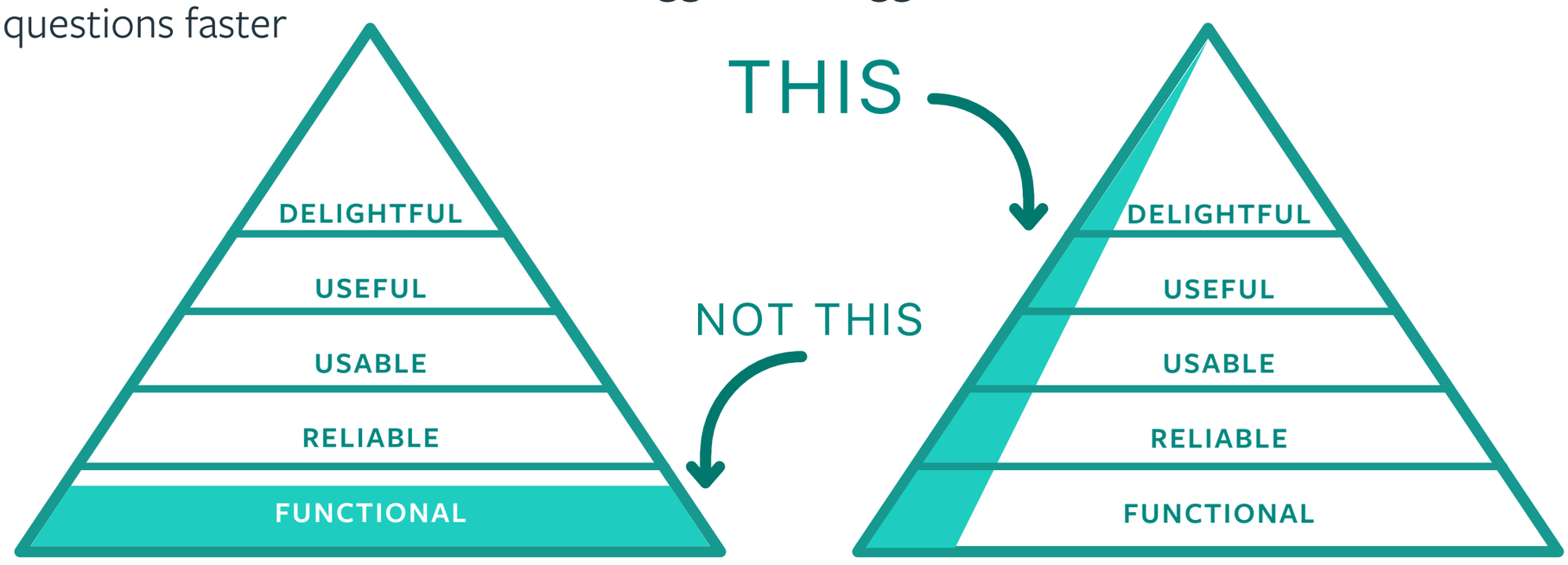
The MVP is the most misunderstood concept in lean product development. Its purpose is to help us learn whether we should continue to build the product or not. Therefore, an MVP is not a delivery milestone, it's a learning milestone.

The MVP is an experiment designed to test our product's value proposition. It's the smallest possible version of our product that enables us to learn the most about our customers with the least amount of effort. Do early adopters of our product find it so valuable that they'd be upset if they no longer were able to use it? Have we found a repeatable way of selling our MVP? The MVP helps us answer these questions faster and with less waste.

An MVP is not

- The first version of the product that we feel comfortable showing our executive sponsors or releasing to market
- A release that we define a priori
- A proof of concept
- A minimum set of features without an accompanying set of business goals and KPIs
- A complete product for internal demonstration purposes
- Always software

The Minimum Viable Product is few features done well. We iterate on the product not by building it one layer at a time, but by building a small pyramid first and then making the pyramid bigger and bigger.



Examples: Minimum Viable Products

Dropbox

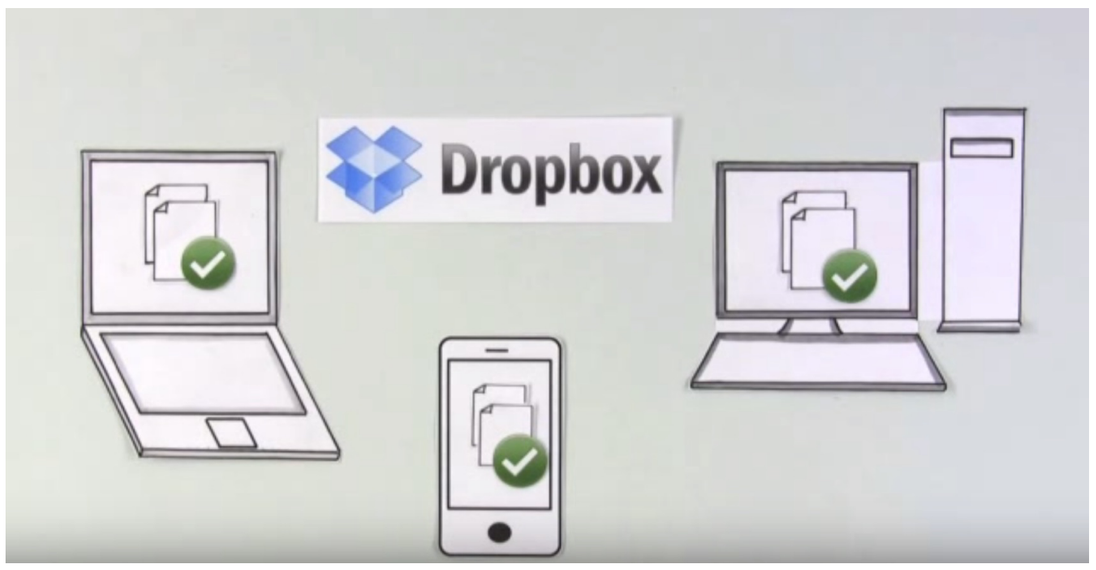
Explainer Video

Dropbox is an extremely popular service for hosting and sharing files, used by 500 million people and 200,000 businesses.

Dropbox's early Leap of Faith assumption was that file synchronization was a problem most people didn't know that they had, but if Dropbox could provide a superior customer experience, people would try it. The team faced two challenges in testing this assumption:

- At the time, Dropbox as a concept was difficult for customers to understand because there wasn't anything quite like it available.
- Dropbox was a technically highly sophisticated product from the beginning and it was impossible to demonstrate it as working software without actually building it.

Instead of investing years in buiding the actual product, the team decided to make an explainer video instead to show how Dropbox works. The video attracted hundreds of thousands of site visits and grew Dropbox's waiting list from 5,000 people to 75,000 people.



Still from Dropbox's explainer video (click to view the video)

Zappos

Wizard of Oz

Zappos is an online retailer with a $2B annual revenue. It was acquired by Amazon in 2009 for $1.2 billion.

When Zappos was founded in 1999, people weren't yet accustomed to buying shoes and clothes online. Founder Nick Swinmurn's Leap of Faith assumption was that selling shoes online was a viable business idea. Instead of buying inventory and creating an online store, he headed to his local mall, photographed pairs of shoes and posted them for sale on a simple website.

When a customer placed an order, Nick would go back to the mall, purchase the shoes and them ship them to the customer. This enabled the Zappos team to learn quickly that people were willing to buy shoes online. They also learned about customer demand and which styles sold best.

Airbnb

Concierge Test

Airbnb is a global accommodation rental service, with a revenue of $900 million.

In 2007, Airbnb's founders wanted to start a business. They could barely afford their rent and decided to offer their apartment as cheap accommodation for design conference participants coming to San Francisco. They photographed their loft, posted the photos online, and shared the link with some friends who were going to the conference who spread the link to their friends. Soon, they had three paying guests. This enabled them to test (and validate) that people were willing to stay in a stranger's home rather than hotel.

Prioritize Features

One of your most important responsibilities as a Product Manager is to prioritize features: decide what to build (or not), and when to build it.

Say No a Thousand Times

People think focus means saying yes to the thing you've got to focus on. But that's not what it means at all. It means saying no to the hundred other good ideas that there are. You have to pick carefully. I'm actually as proud of the things we haven't done as the things I have done. Innovation is saying no to 1,000 things.
– Steve Jobs

Feature ideas can come from many places:

- Anyone on the product team
- Your product sponsors and stakeholders
- Your founders and your board of advisors
- Your customers
- Your competitors

As PMs, we must filter these ideas. It's not possible to do everything, and it's even less possible to do everything at the same time. We need to make sure that the team is only working on features that should be built, and always working on the next highest-value feature. This means we must learn when (and how) to say "no", and when to say "yes". Unless you know when to say no, you'll end up with a lot of tangentially related features, a complex product that no one is really happy with and an overworked product team.

Is It Valuable? Can We Do It?

There are several heuristics you can apply to make informed prioritization decisions. You'll need to consider viability (will it help our business?), desirability (do our users want it?), and feasibility (can we do it?).

You'll find that feature prioritization is part science and part art. You'll need to develop an instinct for what your users want, and then use data and a robust process to back it up.

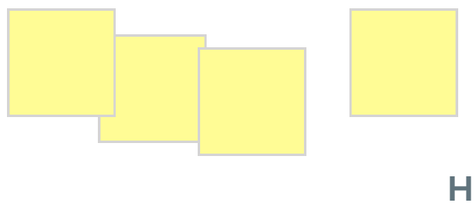
At a minimum, the features we decide to build should:

- Satisfy a user need or desire...
- ...be shared by most users
- Help you make progress against a business goal
- Be possible for the team to build well and ship fast

Example: HomeWiFi Feature Prioritization

HomeWiFi has validated an installer persona and a Minimum Viable Product. They are now ready to start thinking about which features to include in their installer dashboard app.

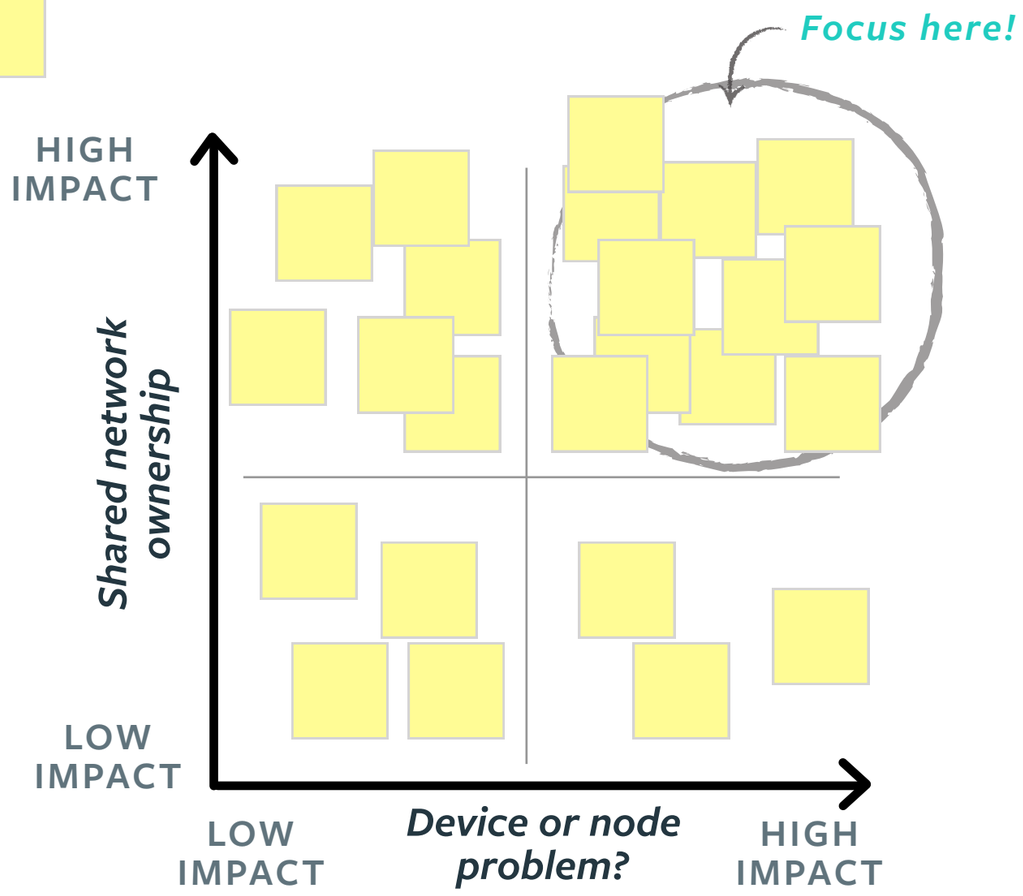
1 Brainstorm feature ideas



2 Organize the ideas based on impact against top user needs

HomeWiFi has learned that installers have two primary needs when it comes to diagnosing network issues:

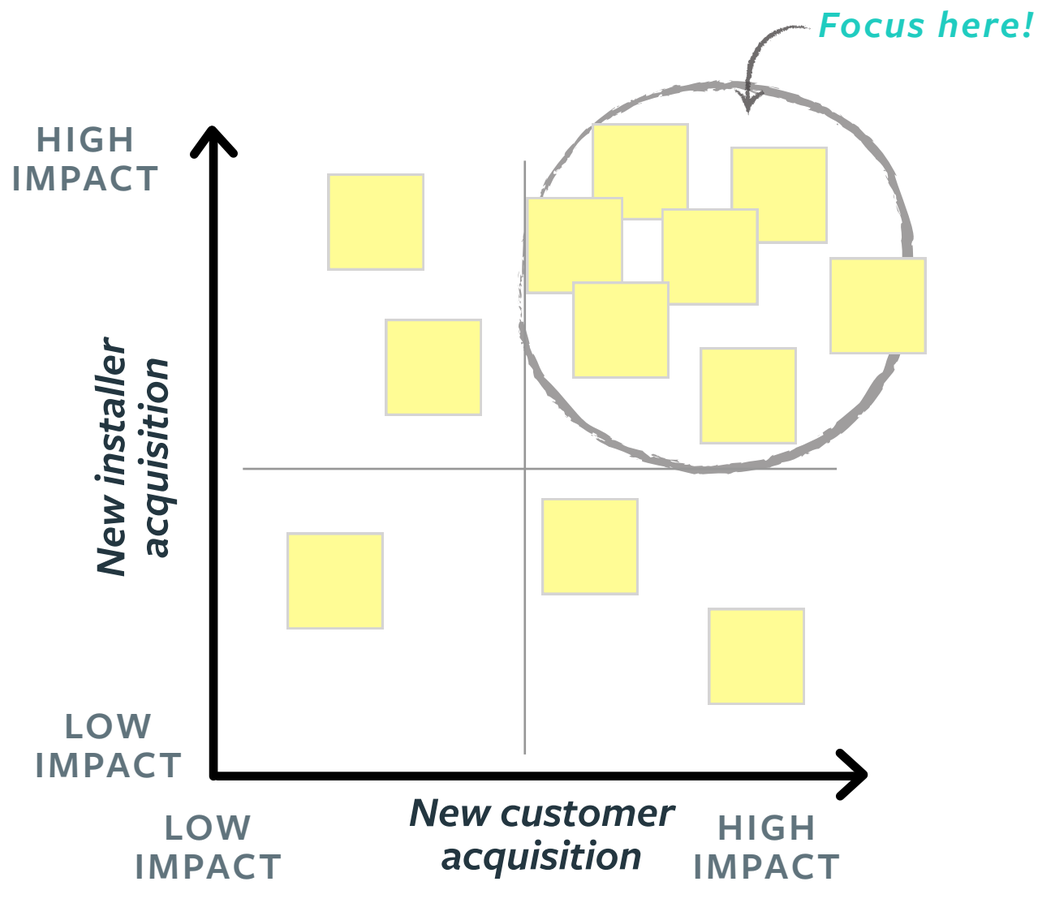
- They need to get reliable access to the customer's network via shared ownership
- They need to know if it's a device connectivity problem or a HomeWiFi network node connectivity problem.



3 Take ideas that have high impact against user needs and organize them based on impact against business goals

To strenghten their position in the home owner market, HomeWiFi needs to expand their footprint and popularity in the installer market. Therefore, their two primary business goals are:

- Grow the number of installers who support HomeWiFi's system
- Help installers acquire new customers



4 Take ideas that have high impact against business goals and organize them based on effort

Start by building features above the horizontal line. These are features that help you address your top user needs and either one or both of your top business goals.

Manage the Backlog

The backlog is where we translate our product vision and strategy into the day to day tactical work of software development.

What is a Product Backlog?

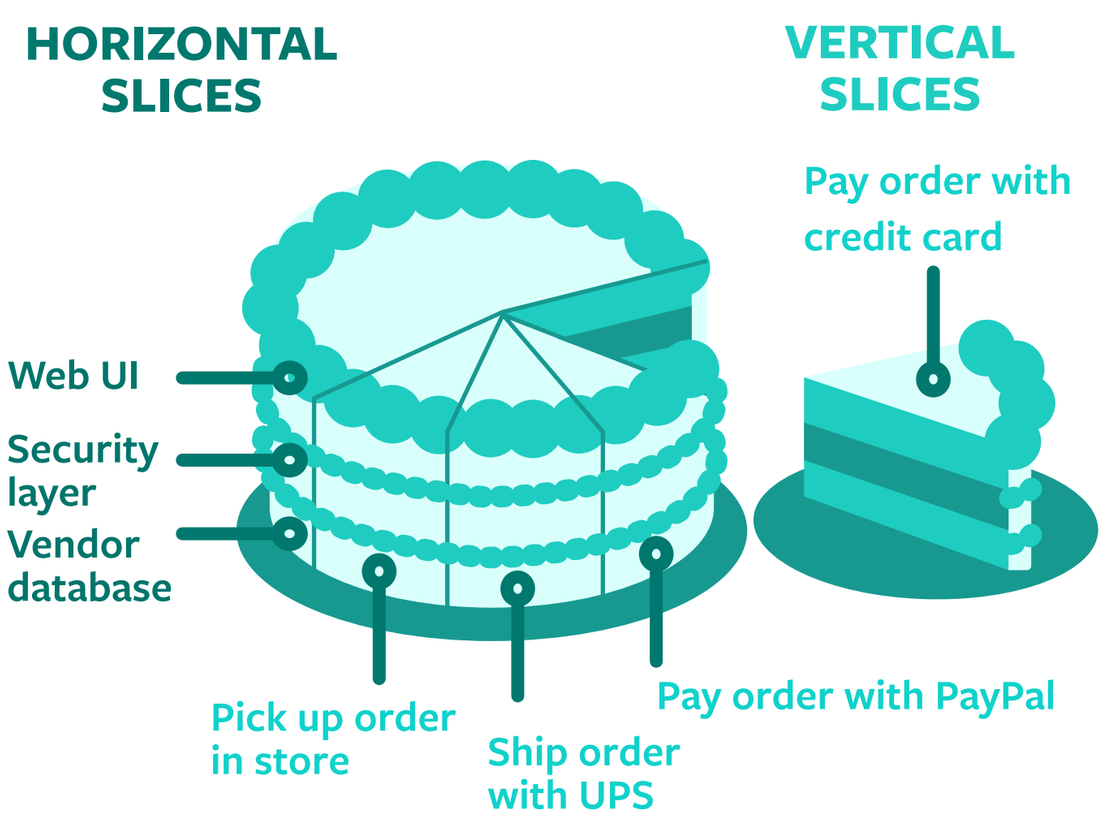
The backlog is a list of development work derived from the product roadmap, organized by priority. The most important piece of work is at the top. A well organized and up-to-date backlog enables the team to build features and functionality efficiently, with a predictable rate of output.

As the PM, you own the backlog. However, you will gather input from all perspectives on the team to inform priority.

Breaking Down the Product

Many teams build products horizontally, breaking the work down along architectural boundaries. For instance, separate teams might build the backend vs the front end layers. The problem with this approach is that we can't ship anything usable to our users until we've finished and connected all the layers.

Agile teams aspire to work in vertical slices, which means that we build the product one small feature at a time. This allows us to frequently deliver new value to users and as well as get their feedback on the new functionality. Frequent user feedback helps us make necessary adjustments sooner and at smaller scale than with big releases.



One vertical unit of development work may encompass GUI, client and backend work – whatever is needed to deliver one unit of value to the user.

These thin vertical slices are commonly referred to as stories. As PM, you will work closely with your team to split your development work into good stories.

The Smaller Slices, the Better

The smaller we can make a story, the simpler it will be to understand, estimate, implement and test. We can more easily predict when features will be ready to ship. Developers can rotate through the codebase quicker and grow the team's bus count. Small stories enable frequent tangible progress which feels great for the team, the stakeholders and the product sponsors, and means that value is delivered quickly to users and the business..

Continuous Prioritization Enables Agility

Agile/XP teams practice continuous prioritization. That is, while we do look a few weeks' ahead to plan our work at a high level, we don't commit to a fixed sprint scope. This is because we may at any time learn about unexpected changes or new information that may make us reevaluate our priorities. Continuous prioritization enables us to maximize our responsiveness to changing conditions and navigate complexity and uncertainty better than if we were to stick to a detailed upfront plan.

Developers will always pick up and work on the story at the very top of the backlog, so the order of the stories is very important. A story's position in the backlog reveals its priority - the most important stories are at the top of the backlog, while the less important stories are near the bottom. It's your job as PM to ensure the order of the backlog represents the latest priority.

Balance Value vs. Quality vs. Constraints

Lean/agile teams must frequently make tradeoffs between value, quality and constraints to successfully respond to change.

Making Tradeoffs Against the Right Goals

Project management and product management are two commonly applied mindsets when it comes to defining software success.

Project management is a tactical role responsible for achieving a specific pre-defined objective by following a plan with a fixed start and end date and a fixed budget. Success is to deliver on time, on scope and on budget. In the context of product development, project management solves for the wrong problems. Product failure is rarely due to schedule slippage or cost overrun. It is due to building something customers don't want or need, or something that doesn't create sufficient business value.

Product requires a different and strategic role: product management. Product management is responsible for defining what the product is and for whom, and how the product will drive business impact. Success is to deliver the right value at the right time.

Value, Quality, Constraints

Product initiatives have three goals:

1. Create customer **value** that drives business impact
2. Build in and maintain **quality** to enable the team to develop at speed forever
3. Deliver within given **constraints**: people, time and scope

Companies often want to keep all three goals fixed. Because change is inevitable, this leads to failure – something unpredictable happens, the team can't realize all three goals and gets burned out trying. Lean/agile organizations build in flexibility by keeping fixed two out of three goals at the most, and making tradeoffs against the flexible goal(s).

Adapted from Jim Highsmith: Agile Project Management

What is Value and Why Does it Matter?

Value is the benefits delivered to customers through the features we build and the business impact created when customers engage with the product. Value is measured by what our customers want or desire, and by the KPIs we need to meet to have a successful business.

To deliver value, we need to do the simplest thing that delights the customer, that is margin-enhancing and that helps differentiate us in the market. This requires us to know what features to build, what features to not build, and to ship new and improved features continuously.

What is Quality and Why Does it Matter?

Quality is defined as reliability and adaptability. Reliability means that our customers have a consistently good experience using the product. Adaptability means that we are able to keep evolve the product at a predictable pace so that it continues to deliver future value. To keep quality high and maintain a steady foundation we need to keep our design and tech debt low.

How Should We Think About Constraints?

Constraints are important but they are not the actual goals of the product. They work as guardrails for the team, establishing clear expectations about delivery. Only one of the three constraints can be fixed. In agile organizations, it's usually time.

Pivotal Tracker

We manage the backlog using Pivotal Tracker, shortened ‘Tracker.’ Pivotal Tracker is the agile project management tool built by Pivotal Labs. It used by more than 500,000 people worldwide.

Why?

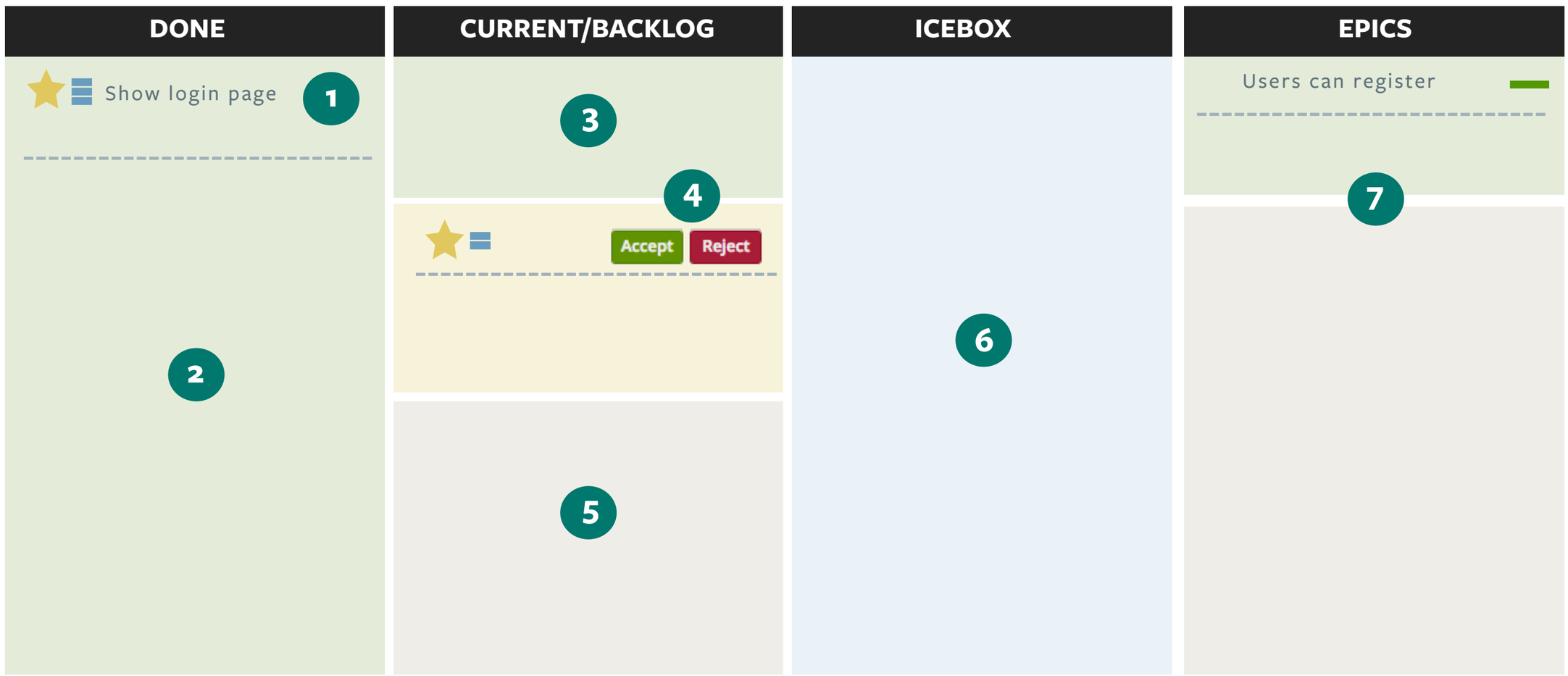
Tracker is a simple, story-based agile project planning tool that allows teams to collaborate and react instantly to real-world changes. It’s based on agile software development methods, but it can be used on a variety of types of projects.

Tracker frees you up to focus on getting things done, without getting bogged down keeping your plans in sync with reality.

How?

Tracker becomes the central repository and historical record of what will be done and a way to look back on development history.

☐ **ACTION:** Ask your Labs PM to help you get set up with Tracker credentials, then go through how to use Tracker together.



1 **USER STORY.** A story is something that a user wants, like “See activity feed” or “Filter product catalogue”. Stories can be grouped together by labels.

2 **DONE.** These stories have been implemented by the development team and accepted by the PM.

3 **CURRENT.** These stories have been picked up by developers and are in various states of implementation. The story status is updated as the story is picked up and delivered. Tracker totals the points for all the current stories into a team velocity.

4 **ACCEPTANCE & REJECTION.** If the functionality matches the acceptance criteria outlined in the story, you should accept it. If the functionality does not meet the acceptance criteria, reject the story and add a comment explaining what is incorrect or missing.

5 **BACKLOG.** The backlog contains stories that are ready to be estimated or worked on. The story in the top is the next story that will be implemented. Stories are prioritized based on what will add the most user value.

6 **ICEBOX.** These are stories that need more thought and detail. The content of the icebox does not need to be prioritized.

7 **EPICS.** These are larger user stories or themes that are too big to be described in a single user story, like “User can register” or “Administrator should see user analytics”.

BACKLOG

The Story Workflow

Agile development consists of a continuous feedback loop. Each story has a workflow from conception to release.

Why?

Agility comes from frequent feedback. Story acceptance, rejection, and release gives you the opportunity to give feedback to and get feedback from your customers, your team, and your stakeholders.

1 **PRIORITIZATION**
The Product Manager prioritizes stories in the backlog.

2 **ESTIMATION**
The team discusses and collectively estimates each story.

3 **STORY START**
Developers pick up and begin work on the next available story.

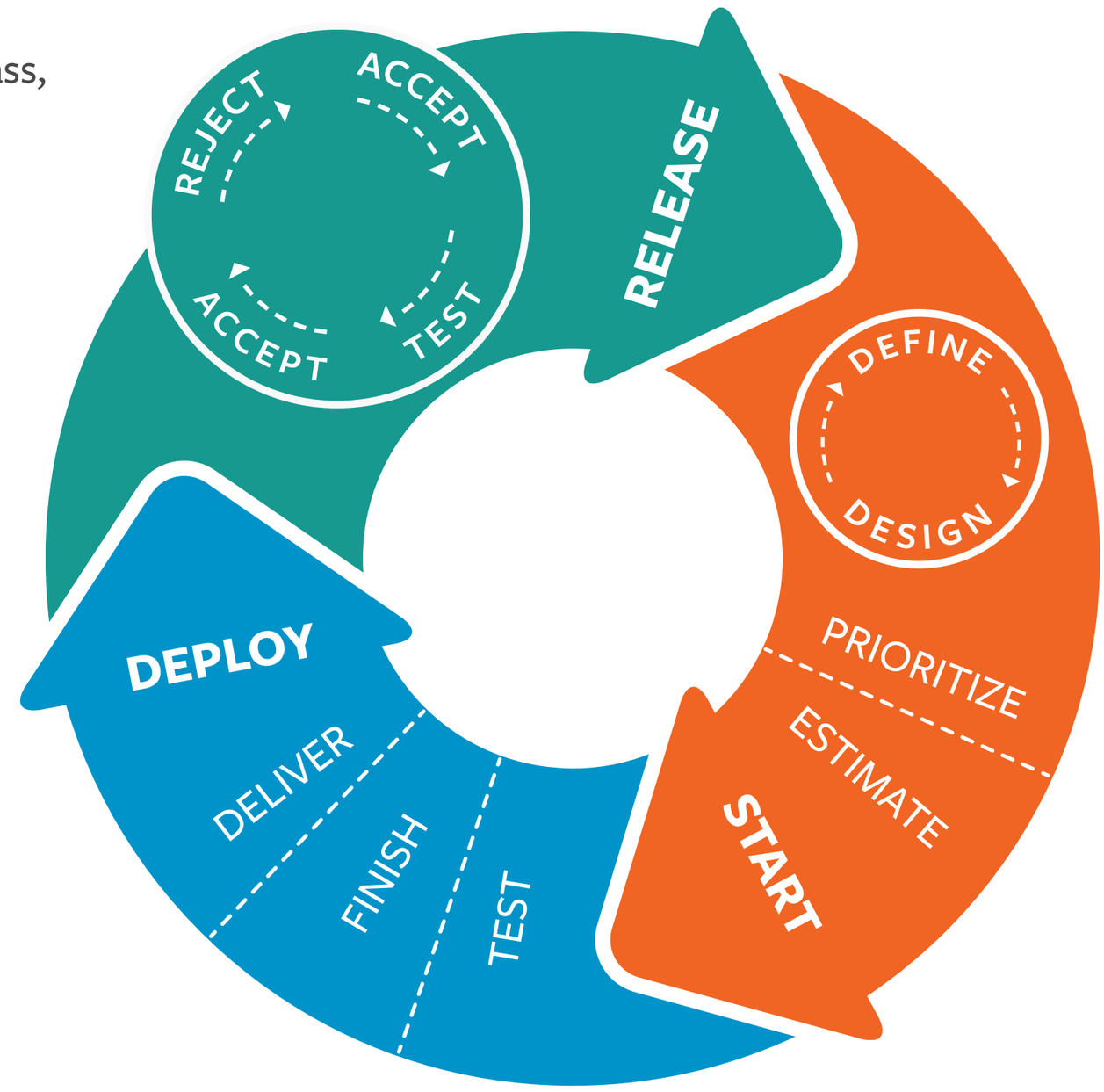
4 **STORY IS FINISHED**
The developers commit all code changes to the project repository, and finish the story.

5 **STORY DELIVERY**
The committed code changes go through Continuous Integration testing. When CI tests pass, the code for the new feature is deployed to the team's acceptance environment, and the story is delivered.

6 **ACCEPTANCE**
The Product Manager reviews delivered stories against their acceptance criteria. If a story is complete, the Product Manager accepts it. If a story is incomplete the Product Manager rejects it.

7 **PRODUCTION RELEASE**
The code for the accepted stories is pushed to the team's production environment, where users can interact with the new features.

8 **... AND REPEAT.**
Based on user feedback, input from the business and what we learned from our previous product release, the Product Manager determines what to prioritize next.



Plan With Stories

Knowing what to do next is one of the agile Product Managers most important skills. An effective product plan is made up of many small, independent user stories. This allows you to plan and easily respond to change.

The Product Manager As Planner

The Product Manager is accountable for planning the work efficiently. This means:

- Taking a large scope of work and breaking it down into manageable steps to complete it
- Allowing focus on completing the next priority in the list
- Seeing progress and the big picture

Knowing how to prioritize, estimate and organize stories and track team velocity is fundamental to good project planning.

All Stories Must be Prioritized

Guided by input from users, stakeholders and the development team, the Product Manager weighs user value, business value, development risk and dependencies against each other to determine the priority of a story.

A story's position in the backlog reveals its priority; the most important story is at the very top while the least important story is at the bottom. This makes it obvious which task the team will work on next.

Stories' priorities can change as business priorities change.

User Stories Must be Estimated

All user stories need to be given a point estimate before developers can work on them. New stories are discussed in the weekly IPM. Once everyone understands the purpose of the story and agrees on the simplest way to accomplish it, the team can agree on its size.

Knowing the relative complexity of stories helps the Product Manager prioritize and set expectations with stakeholders.

For instance, you can decide to prioritize a big story or three small stories – the amount of work would be about the same but in the first case you ship one larger feature whereas in the second case you ship three smaller features. It is important that the team focuses on complexity rather than effort as the points are ultimately a planning tool and not an exact time estimate.

Bugs and chores are not given point estimates. The idea is that these two story types emerge over time, and while they do take time to address, they're an ongoing and fairly consistent cost.

Larger Stories Are Grouped Into Epics

A group of stories that represent a larger features is called an epic. Epics convey the overall big picture priorities.

Velocity is a Measure of Average Delivered Story Points

Average velocity is an indicator of how many points a team can be expected to deliver in a given iteration. It is calculated by averaging the points delivered in previous iterations. Only completed stories count toward velocity which is why regular story acceptance is important.

When your team gets good at building up features with small, consistently-sized stories, your team gets better at delivering about the same amount of work each iteration. With lower volatility, your team's velocity becomes a more useful planning tool. The iterations planned into your backlog reflect past reality and are useful predictions.

Respond to Change

When building software it's impossible to gather all the requirements up front. Responding to change requires knowing how to re-prioritize, estimate & organize stories.

Write User Stories

A user story is a short, simple description of a feature told from the perspective of the person who desires the new capability, usually a user or customer of the system.

Why Do We Care About User Stories?

User stories are designed to explain the who, what and why of the smallest incremental feature that can be added to a product. They are written from the user's perspective, explain incremental business or user value, and act as light weight requirements documentation for a development team. A story is a placeholder for a conversation. After a story is completed, it becomes a view into our conversation history and the decisions we've made.

Who Writes stories?

While anyone on the team can contribute stories, it is the Product Manager's responsibility to maintain a healthy backlog. The Product Manager does this by writing and prioritizing enough user stories for two weeks of development work.

How?

1. **TITLE.** The title should be short and descriptive.

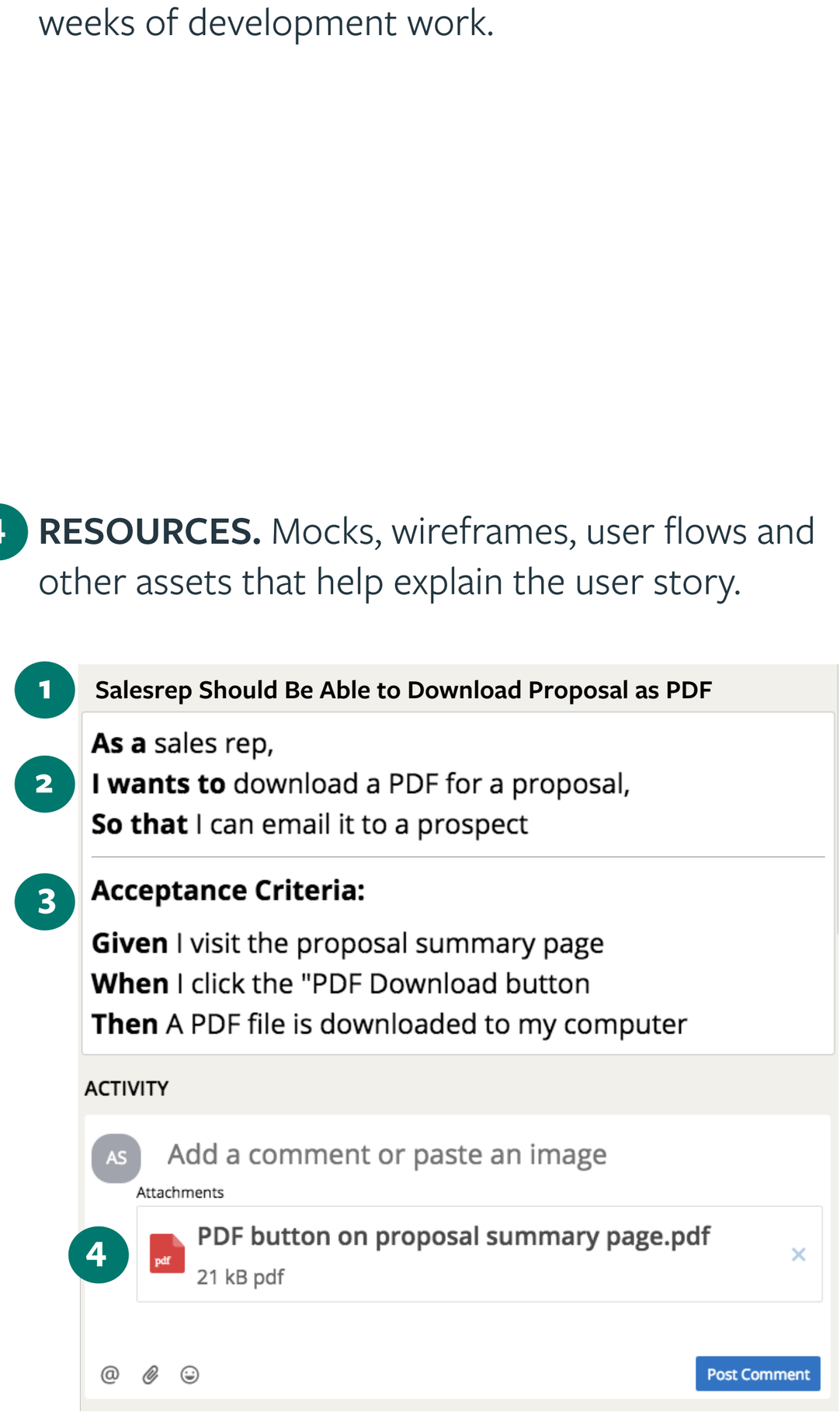
2. **DESCRIPTION.** The description should explain who wants the functionality, why and to what end. This typically follows the form:

AS A	[TYPE OF USER]
I WANT	[SOME GOAL]
SO THAT	[BENEFIT]

3. **ACCEPTANCE CRITERIA**. The acceptance criteria are a list of scenarios the Product Manager will use to test that the story has been completed and is also used as a starting point for the developer's tests. This typically follows the form:

GIVEN	[SOME CONTEXT]
WHEN	[SOME ACTION IS CARRIED OUT]
THEN	[SOME CONSEQUENCES]

4. **RESOURCES.** Mocks, wireframes, user flows and other assets that help explain the user story.



An example user story in Pivotal Tracker

☐ **ACTION:** Read about the INVEST mnemonic with your Labs PM to better understand the What and Why of writing backlog stories.

Example User Stories: Good vs. Bad

A backlog of well-written user stories enables the team to focus on doing the work rather than trying to define the work.

EXAMPLE: GOOD USER STORY

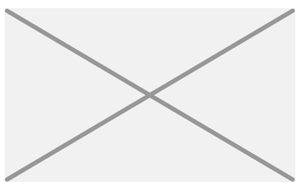
Sales Rep should be able to download a proposal as a PDF

DESCRIPTION

As a Sales Rep
I want to be able to download a PDF for a proposal
So that I can email it to a prospect

ACCEPTANCE CRITERIA

Given I visit the proposal summary page
When I click the "PDF Download" button
Then a PDF file is downloaded to my computer



mock_of_pdf_button_on_proposal_page.png
(120 kb)

What Makes This a Good User Story?

- The title is clear and descriptive
- The user is clearly identified
- The story has a clear beginning and end
- The acceptance criteria satisfy the user's goals
- There are resources attached that describe all the non-obvious details that are important to the user and the business
- The user story represents the smallest amount of verifiable functionality that provides incremental value

EXAMPLE: BAD USER STORY

Proposal as PDF

DESCRIPTION

As a user
I want to be able to save PDFs for a proposal, see all PDFs generated for that proposal, and download them

ACCEPTANCE CRITERIA

- Proposal page includes a list of PDFs generated
- User can generate additional PDFs
- User can download the PDFs from the list

What Makes This a Bad User Story?

- The title is clear and descriptive
- The user is clearly identified
- The story has a clear beginning and end
- The acceptance criteria satisfy the user's goals
- There are resources attached that describe all the non-obvious details that are important to the user and the business
- The user story represents the smallest amount of verifiable functionality that provides incremental value

Other Story Types in Tracker

Chores, bugs, and release markers enable you to capture work that needs to get done but doesn't provide direct value to the user or the business.

Chores, Bugs and Release Markers Are Stories, Too

Just like the user story, these story types represent concrete tasks or deliverables. However, because they do not provide direct business or user value, they do not get estimated (more on this in "Plan Iterations").

RELEASE MARKER

Releases are milestone markers and allow your team to track progress towards concrete goals. For example stakeholder or investor demos, software launches, etc. It's possible to specify target dates for releases. The Product Manager decides how to organize the backlog into releases.

CHORE

A chore is a story that is necessary but provides no direct, obvious value to the user.

For example:

- Setup new domain and wildcard SSL certificate for test environments
- Evaluate tools for system troubleshooting
- Exploratory testing. This is often referred to as a "Charter".

Chores can represent "technical debt" and/or points of dependency on other teams. Chores are not estimated, as they do not contribute business value directly. Developers, often in partnership with the PM, create chores for the backlog.

BUG

A bug is a defect in a feature that has already been accepted, regardless of when it was accepted. You should not use bugs to detail new features and functionality.

For example:

- Price should be non-negative
- Login button doesn't work

Bugs do not have points because they are directly related to features that have already been delivered. A bug description should include steps to recreate the bug such that anyone, with minimum context, can see the bug themselves.

Anyone on the team can create a bug. It's up to the Product Manager to prioritize it.

Know Your Story Markers

USER STORY. The 'who', 'what', and 'why' of a new feature.

STORY POINTS. The estimated relative complexity of a story.

CHORE. Tasks that are necessary but don't add direct or obvious user value.

BUG. Tasks that will resolve unintended behavior.

RELEASE. Project milestone.

Run the Iteration Planning Meeting

The Iteration Planning Meeting (IPM) is core to an agile development practice and provides the opportunity for product manager(s) to communicate the vision for the upcoming iteration.

Frame the conversation. Establish the theme of the next week(s), and tie it back to product and project goals. For example: "Just to reiterate, last week we focused on making our site more mobile-friendly, and this week we're going to revisit the sign-up flow with some improvements. This will help us convert more people coming in from our new mobile marketing campaign."

Clarify the stories. Start by looking at any stories still in progress from the last iteration, along with any pertinent information about features just finished. This should be a quick (<10 minutes) process to jog everyone's memories and ground them in the work that's coming up.

Starting at the top of the backlog, step through each story. Talk through the user-facing value for a feature, ensure that any comps, wires, assets, flows, data, etc. are attached to the story. Clarify any acceptance criteria. The goal here is to be crystal clear on when a feature is "done done".

Complexity check. If it doesn't already have an estimate, each developer that could work on a story should roll on the points to deliver. If the implementation is not clear, they should have time to talk through approaches. That said, their role is to nail down a level of complexity, not pin themselves to a specific technical implementation. Based on the estimate size or developer feedback, stories can be nominated for merging or splitting up. If that's the case, capture the pieces of work as placeholders and update with details post-IPM.

Pay down debt. A healthy development process will incorporate refactors and tackle technical debt in concert with new user value. In addition to explicit tech debt chores, PMs and developers should look for opportunities to wrap this work into feature development. For example, if a story calls for adding a field to an existing form you should consider also cleaning up the logic that delivers form validation errors.

Two iterations, max. Tracker will chunk stories into iterations based on the team's velocity. You should only step through stories until you've got two iterations worth of estimated work. Keep the visibility to two weeks to cover for quicker than intended delivery of features, and limit the IPM to a reasonable amount of upcoming work. It's taxing to keep the mental inventory of features in your head; sticking with the short term future focuses everyone on the team around tangible new features.

Keep it short and sweet. Once you hit an hour of IPM, people zone out and business owners get antsy about the emails they're missing (or worse, they whip out their phones). When you have a large team, it's especially important to play time cop. If you don't have a healthy backlog and find yourself with a lack of new work, end the meeting. It's far better to hold an emergency IPM two days later than to suffer the pain of making up stories in real time!

Decide When to Ship Software

The more often we ship, the more often we deliver concrete value to our customers and validate our product strategy.

Ship Early and Often

The ultimate goal of any product team is to ship the right working features to users. It's not until our customers interact with our features that we truly know whether we built the right features, and designed and implemented them right.

User testing without putting software in the real hands of active users is very limited in terms of what we can learn. Working software can help answer questions like "Have we built enough? Are we focusing on the most important feature first? Are we likely to hit our KPIs that the business believes we're going to hit?"

We want to keep this feedback loop with our users as short as possible so that we can continuously validate our product direction. Should we go deeper on features we have or go wider on new features? Based on user value, business objectives and technical constraints, we should figure out what the smallest thing is that we can release to help collect data on what our users find valuable to inform our work.

We also want to make our releases as easy and cheap as possible. If releases are risky, expensive and cumbersome, we will push them off. If releasing is a big, risky and expensive undertaking, we'll only be able to do it occasionally. A repeatable, regular, successful, reliable and cost-effective release process is critical to our ability to continuously deliver small chunks of value. Frequently releasing thin slices of our product enables us to easily roll back any changes, and debug issues quickly.

WHAT'S THE PATH TO PRODUCTION?

Be proactive about understanding and satisfying the requirements to reach the product environment in your organization. This may take longer than expected and become a blocker for the team.

Can We Ship?

Because our ability to ship depends on our delivery infrastructure and the quality of our code, "can we ship" is a technical decision. Our ideal state is to always be able to ship, so that we can release new features when it makes sense to do so from a business perspective.

Our ability to ship is impacted by

- Code quality
- Test coverage
- Our knowledge of what's needed to push code
- Story size and quality

Should We Ship?

The decision to release should come from the business. We need to weigh the user value we can deliver right now against the cost and risk of shipping. We do this by gathering input around:

User needs. How urgently do users need a new feature or refined feature design?

Stakeholders. Are marketing, sales, customer service and other teams ready to support the features we want to ship?

Other product teams. If we depend on other team's services being in production before we can release, we need to make sure those services are ready for us to use.

Release size. The bigger, the riskier.

How manual our process is. The more manual, the more expensive.

Delivery infrastructure. How easy it is to push code to production?

Help Establish Sustainable Pace

To deliver value fast forever, we need to make sure we're running at a comfortable marathon pace rather than rushing through a series of high-intensity sprints.

Let's go Fast, Forever

Sustainable pace is a core tenet of XP. It states that the team should aim to establish a work pace that they can sustain indefinitely. This enables us to build better software, ship it in a predictable manner and do so with regularity.

You'll notice that much in the environment at Pivotal reflects our belief in sustainable pace. The entire office will be present for office-wide standup shortly after 9 am, jointly take lunch for an hour at 12:30 pm, and leave work at 6:00 pm sharp. Morning breakfasts at Pivotal fuel people for the day ahead, and regular breaks clear our heads between bursts of high-focus activities.

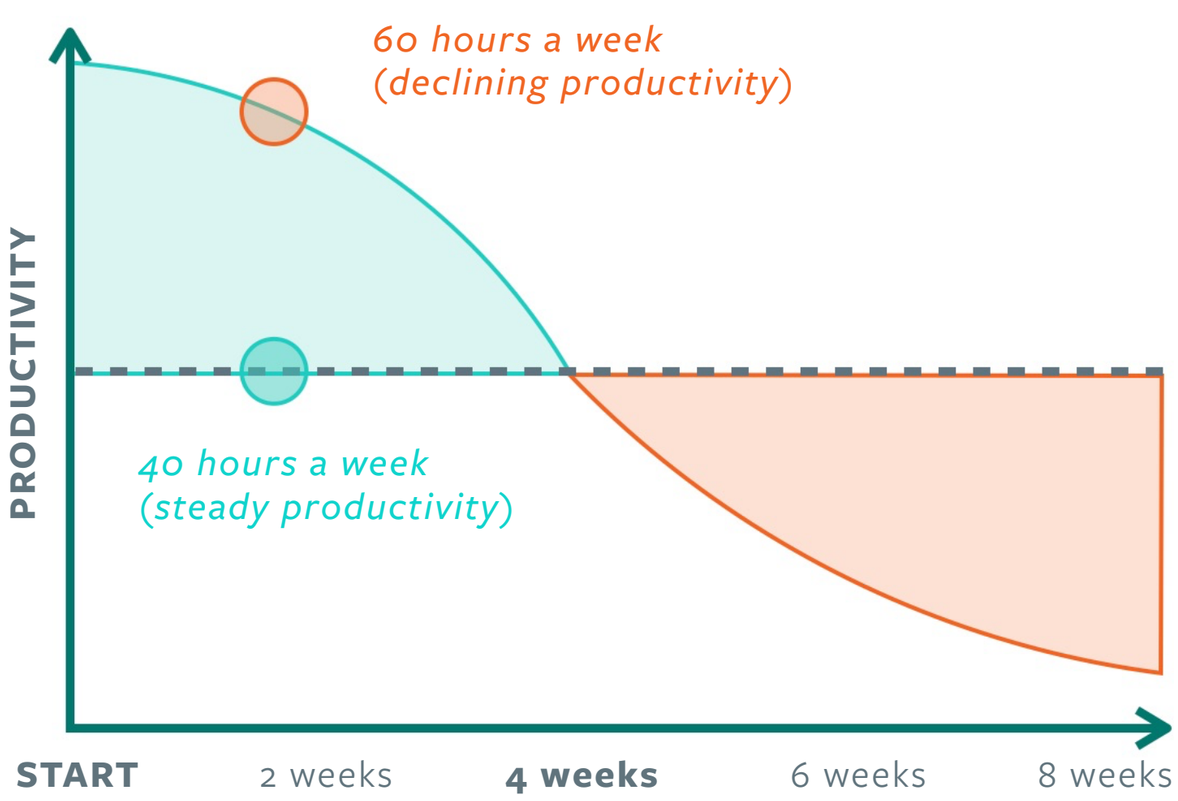
More Hours, More Problems

We believe that building a product is a marathon, not a sprint. We don't do "crunch time" at Pivotal. We believe that putting in significant overtime is detrimental to work quality, team productivity and team health.

Willpower is a depleting resource. If we don't replenish it by sleep, our discipline and work suffer. In the long run, long hours lead to burnout and poor decision making. A developer who stays up late to code a feature will realize the next morning that they'll need to rewrite it all. Even if the code is good, working late without the rest of the team reduces shared ownership of the project by creating knowledge silos where one developer makes decisions without consulting the team. Working 8 hours a day at full intensity and then leaving work concerns at work will produce better results than working extra hours on a regular basis.

We believe that Labs teams get more done by identifying and minimizing any non-productive activities during the work day, allowing the team to focus on value-generating activities for the majority of the day. Hours spent don't matter so much if we focus on maximizing the value we produce. Many Labs first-timers report being exhausted after their first week here–that's what happens when you get an uninterrupted 8 hours a day to work!

PMs play a critical role in sustainable pace by prioritizing the right product and business goals, and communicating with the right stakeholders to ensure that whatever is delivered isn't a surprise. This means regular demos, stakeholder checkins, maintaining an up-to-date product roadmap, and creating consensus with the business to avoid scope creep and 'crunch time'. PMs should also take initiative on efficiency-improving activities such as removing team blockers, eliminating unproductive meetings, and boosting team morale. Think outside the box for how you can contribute to sustainable pace and team happiness – no team ever complains when their PM surprises them with donuts!



Communicate Effectively

To launch successful products, you need to build a shared understanding with your team, your stakeholders and your sponsors of the What, When, Who, Why and How of the product and each iteration of it.

Strong Communication Skills Help You Lead Without Authority

You will be championing your product within your organization, making trade-offs with stakeholders and other product teams, and facilitating daily tactical decisions with your own team. You'll need to inspire trust, as well as motivate and influence people to help you and your team launch a successful product to market. The strength of your communication skills will determine how well you'll be able to develop the credibility and strong relationships needed to do so:

Stakeholders. Functional teams like legal, security, sales and marketing want to know that you understand their needs and how your product supports their goals. For instance, marketing will want to know how your product will drive leads, and sales will want to know how your product will help close those leads.

Sponsors. Executive sponsors want to know how your product is increasing the bottom line. Share with them the strategic objectives and KPIs you're working towards. Document and share the business impact the product is having.

Core team. Physical co-location creates ideal conditions for short feedback loops. Make sure you are responsive to your team's communication needs by being present and available. Share incoming information from sponsors and stakeholders quickly and as appropriate – shield the team from information overload while making sure they have the information they need to do their work.

Others. You may be working very differently from other teams in your organization. Help them understand how being lean/agile/user-centered impacts the cadence, needs, health and success of your product and your product team.

Tips For Effective Communication

Tell a story. Why does the product exist? Why does it matter? A compelling product story will make the product feel real and create an emotional bond to it, which will make people invested in its success. Use anecdotes, user journeys, prototypes, videos and other artifacts to help tell your product's story.

Explain the big picture. Share the vision, strategy and roadmap. Help everyone involved understand what outcomes you're working towards and how you'll know you've succeeded.

Be believable. When you communicate product decisions, make sure you do so with logic, empirical evidence, enthusiasm and a focus on driving business impact by satisfying customer needs and desires. Let product designers and anchors communicate the rationale behind design decisions and technical decisions.

Adjust the message and content to your audience. Tell them what they need to know. Only share the information they need to make a decision or take action.

Be honest. When new information arrives that necessitates a shake-up of the product plan, you must have the courage to recognize any issues, communicate new data and insights back to the team and stakeholders, and facilitate the creation and implementation of a new plan.

Be proactive. To build trust with stakeholders and collaborators, share updates with them proactively. Don't wait till the last minute or when you need something to reach out.

Listen well. Be as interested in learning from others as you are in sharing with them. Active listening will help you build stronger relationships and may also help you gain new insights that improve your product.

Glossary

A

Agile	A group of software development methods based on iterative and incremental development, in which requirements and solutions evolve through collaboration between self-organizing, cross-functional teams. It is a conceptual framework that promotes well planned, small iterations throughout the development cycle.
Anchor Role	An experienced developer who, in addition to coding full-time, leads the technical aspects of the project from from start to end. The anchor acts as a resource for the rest of the development team for technical and non-technical issues.

B

Backlog	The backlog is a list of prioritized stories that make up the planned work for the current iteration. Stories can be added and removed from the backlog and re-prioritized as necessary during an iteration.
Balanced Team	An autonomous team that has people with a variety of skills and perspectives that support each other towards a shared goal. The team values cross-disciplinary collaboration and iterative delivery.
Blocker	A situation or issue, brought up at project standup, that is delaying or preventing project progress. The team self-organizes to resolve the blocker and when it cannot be unblocked by the team, it can be escalated to the Client Liason and/or stakeholders.
Build, Measure, Learn	A core component of Lean Startup Methodology. Build-Measure-Learn is a feedback loop in which we first figure out what problem needs to be solved, then build and test the smallest possible solution.

C

CI	CI stands for “Continuous Integration”, it is a server which is a dedicated machine that runs a project-specific defined set of tests periodically.
CD	

D

Design Crit	A feedback session Designers run to get feedback from fellow Designers and Product Managers.
Design Review	A weekly team meeting to get an update on the progress of design and research. The goal of a design review is for a designer to leave the meeting with a clear list of updates to the designs and workflows they’ve presented based on feedback from developers, PMs, and client stakeholders on feasibility, business value, brand, priority, and scope.

Design Studio	A solution brainstorming activity a Designer leads to gather ideas in the form of sketches.

E

Empathy	Empathy is an ability to understand what other people are thinking and feeling. We hire for empathy and compassion because it enables us to be kind and effective collaborators. Good collaborators build happier teams.
Epic	An epic is a collection of stories that make up a larger product release, feature set, or development focus. Epics are useful for prioritizing groups of stories against other groups of stories.

I

Iteration	A 1-week planning cycle. Planning and development is iterative. Because we are constantly coding and testing, the products we build are always ready to go live. This iterative process allows for changes as business requirements evolve. Daily and weekly software builds provide constant validation that the software meets the business requirements. You always have complete control of the product and the timeline.
IPM	IPM stands for Iteration Planning Meeting. A weekly meeting at the start of an iteration, during which the team reviews the upcoming stories in the backlog, ensures the backlog is full for the next two or three iterations, confirms the prioritization of stories, and estimates any unestimated stories using a point system.

L

Lean	Lean is the practice of build products that deliver the most value for your customers and minimizing waste by systematically identifying assumptions and validating with actual users.

M

MVP	A Minimum Viable Product is the cheapest, fastest, simplest thing that can help validate or invalidate hypothesis about customer behavior.

P

Pair Programming	Pair Programming is the practice of having two developers work together at the same computer to complete each task. At Pivotal Labs we pair all of the time. This practice of focusing two minds on the same challenge leads to better decisions the first time around, fewer knowledge gaps, and continual implicit training and knowledge transfer. Pairing results in fewer defects, better code, and ultimately much more sustainably efficient development. As pairs rotate, knowledge is spread rapidly through the team, avoiding silos of knowledge and allowing for team growth.
Persona	A model of a user of your product/service based on similar needs, goals, context, and tasks of many actual users. Developed through user research. Used to gain and retain empathy for the users we are solving for.
Pointing	In order to measure a story's complexity, Pivotal uses a point system of 0, 1, 2, 3, 5, and 8 points. During an IPM, developers will discuss and agree on a point estimate that reflects the story's complexity to the best of their understanding. Points do not reflect a measure of how much time a story will take to complete since time is especially difficult to estimate and can vary based on external factors. Measuring relative complexity is easier, and allows for much more consistent measures of team progress.

Pivot or Persevere	A pivot is a change in strategy, not in vision. To pivot means to change your business model, product, or target market when you learn based on feedback from the market that your current plan isn't working. To persevere means to continue to follow your plan based on feedback. You should regularly revisit your strategy in a "pivot or persevere" meeting.

R

Rapid Prototyping	The quick and early development of a small-scale prototype used to test out certain key features of the product design. Useful for learning quickly and cheaply.
Risk (product)	Probability of wasting time and resources on features that don't provide business or user value.
Retro	This is a meeting that provides a team to give positive and negative feedback to each other in a structured way that yields action items. Team members will each identify aspects of the previous week that went well, and aspects that went poorly. We typically reflect on issues ranging from technical choices, to inter-role communication, to working environment. These thoughts are then grouped into themes and the team brainstorms and assigns action items to remedy any issues. Sometimes each Retro can begin with a review of last week's action items to improve accountability.

S

Service Blueprint	A diagram of a user's path through an experience, including physical, in person, and screen touchpoints. The diagrams are key in communicating a complex flow, especially if there are multiple interactions spread over several days, weeks, or even months.
Standup	A daily short meeting (usually first thing in the morning) to discuss what was accomplished the previous day, share any info that is valuable to the entire team, ask for help, and determine pairs for the day. The meeting is meant to be as short as possible and any discussions that only involve a subset of the team are moved into separate meetings.
Stakeholder	An individual who stands to gain or lose from the success or failure of a product/feature/solution. They understand and represent business interests and/or functional teams. They may also possess valuable knowledge about customers and end users.
Style guide	A document that serves as a guide for visual design elements. More specifically, a living style guide is a living document of code, which details all the various elements and coded modules of the application.

T

Test-Driven Development	Test-Driven Development (TDD) is a software development process that relies on the repetition of a very short development cycle: developer writes an (initially failing) automated test case that defines a desired improvement or new function, produces the minimum amount of code to pass that test and finally refactors the new code to acceptable standards.

U

User-Centered Design	An approach to product design where the user is put at the front and center of every design activity. User-centered design (UCD) is a way to de-risk the product early and often. The team involves representative end-users throughout the process of discovering, defining, design, developing, delivering and optimizing the product to test whether the features and user experience are useful and usable.

V

Velocity

Velocity is a measure of how many points a team can be expected to deliver in a given iteration. It is calculated by averaging the points delivered in previous iterations.

Volatility

Volatility is a measure of how variable a team's velocity is.

X

Extreme Programming (XP)

Extreme Programming (XP) is a flavor of agile software development. Elements of XP include: programming in pairs, automated testing of all code, avoiding programming of features until they are actually needed, simplicity and clarity in code, expecting changes in the customer's requirements as time passes and the problem is better understood, and frequent communication with the customer and among programmers.

Reading List

Check out these books, videos, and articles to dive deeper into concepts and tools.

Lean

Lean Analytics
by Ben Yoskovitz and Alistair Croll

The Lean Enterprise
by Trevor Owens and Obie Fernandez

The Lean Startup
by Eric Ries

Lean UX
by Jeff Gothelf and Josh Seiden

Running Lean
by Ash Maurya

The Startup Way
by Eric Ries

Making Sense of MVP
by Henrik Kniberg

Agile Development

Extreme Programming Explained
(*Especially Chapters 1-7*)
by Kent Beck with Cynthia Andres

Agile Product Ownership in a Nutshell
by Henrik Kniberg

INVEST (mnemonic)

User-Centered Design

The Design of Everyday Things
by Don Norman

Just Enough Research
by Erika Hall

Practical Empathy
by Indi Young

Talking to Humans
by Giff Constable

Satisfy The Cat
by John Boykin

Pivotal Tracker

Getting Started with Pivotal Tracker
by Pivotal

Other

Balanced Team
by Janice Fraser

Thanks for reading! If you have any questions or feedback about what you've read, please share with us at: **client-pm-playbook@pivotal.io**.

This is v 2.0 of the Pivotal Labs Client PM Playbook.