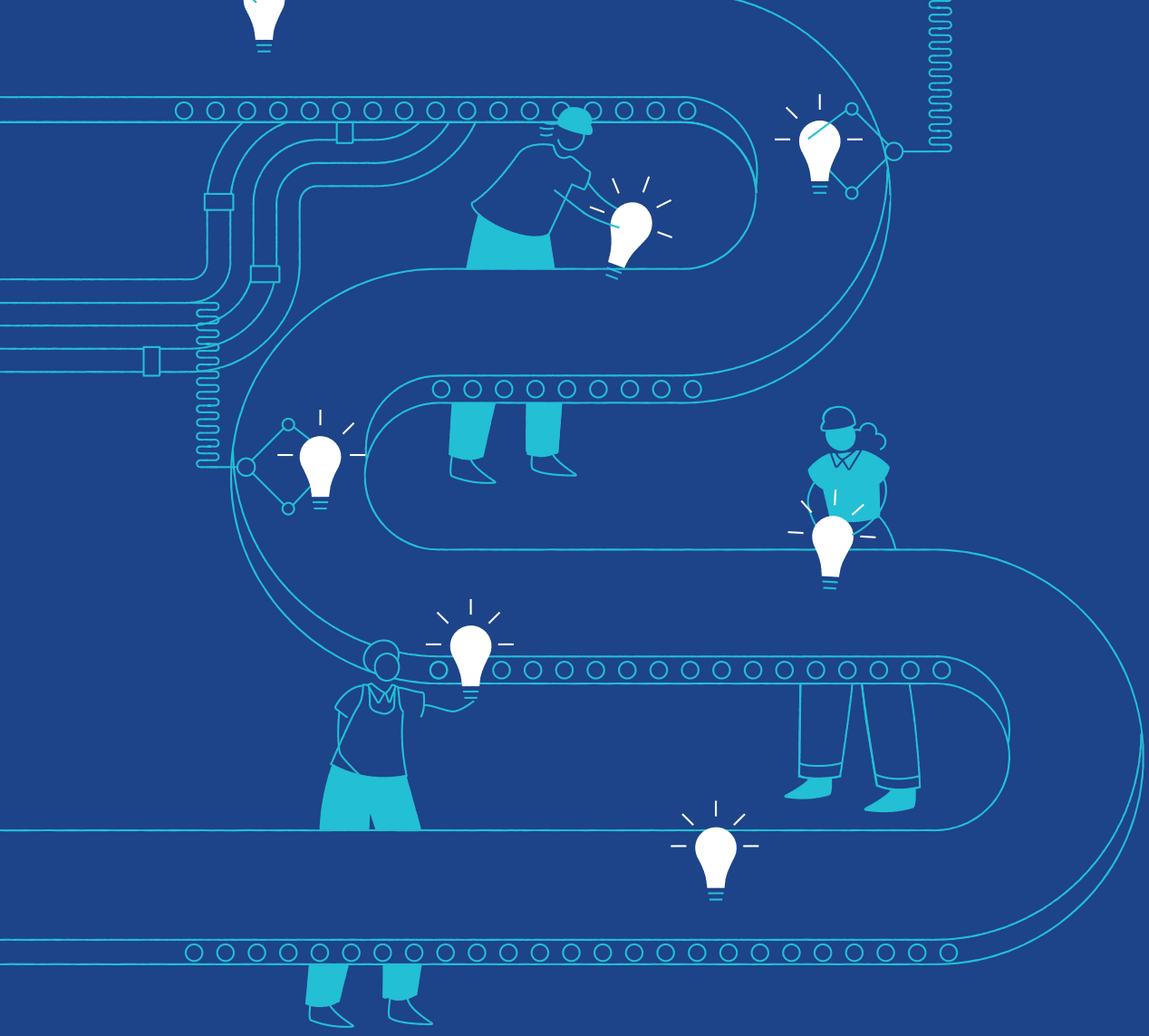


# VMware Tanzu Labs Design Guide

## Agile Meets User-Centered Design

# Contents

# Design is part of our DNA

For years, we've helped our clients build products and scale their design practices. In the process, we've learned that when curious and diverse teams center around the needs of the user, they create a culture of agility that thrives in the uncertain and ever-changing landscape of technology.

Our approach to design democratizes the most valuable parts of the craft, bringing others along for the innovation journey. We've combined the best ideas from user-centered design (UCD), design thinking, and Lean to create high-functioning Agile teams that build great products, quickly, using an Agile UCD approach.

At VMware Tanzu Labs, we use design in four ways:

- pairing with our clients
- building products
- developing internal tools and processes
- supporting humanitarian organizations and charities

This guide explains the unique product design principles we've refined over the years to transform enterprise teams. It's not an instruction manual; rather, our goal is to share how we think with the greater design and technology community.

# The current enterprise design landscape

The rapid growth of technology-enabled products and services continues to alter the way we interact with the world. There's never been more choice for customers. New products and services constantly compete for their attention. But of the more than 30,000 new products introduced every year, 95 percent fail. Responding quickly to customers' needs and maintaining their engagement is the only way companies can survive and stay relevant.

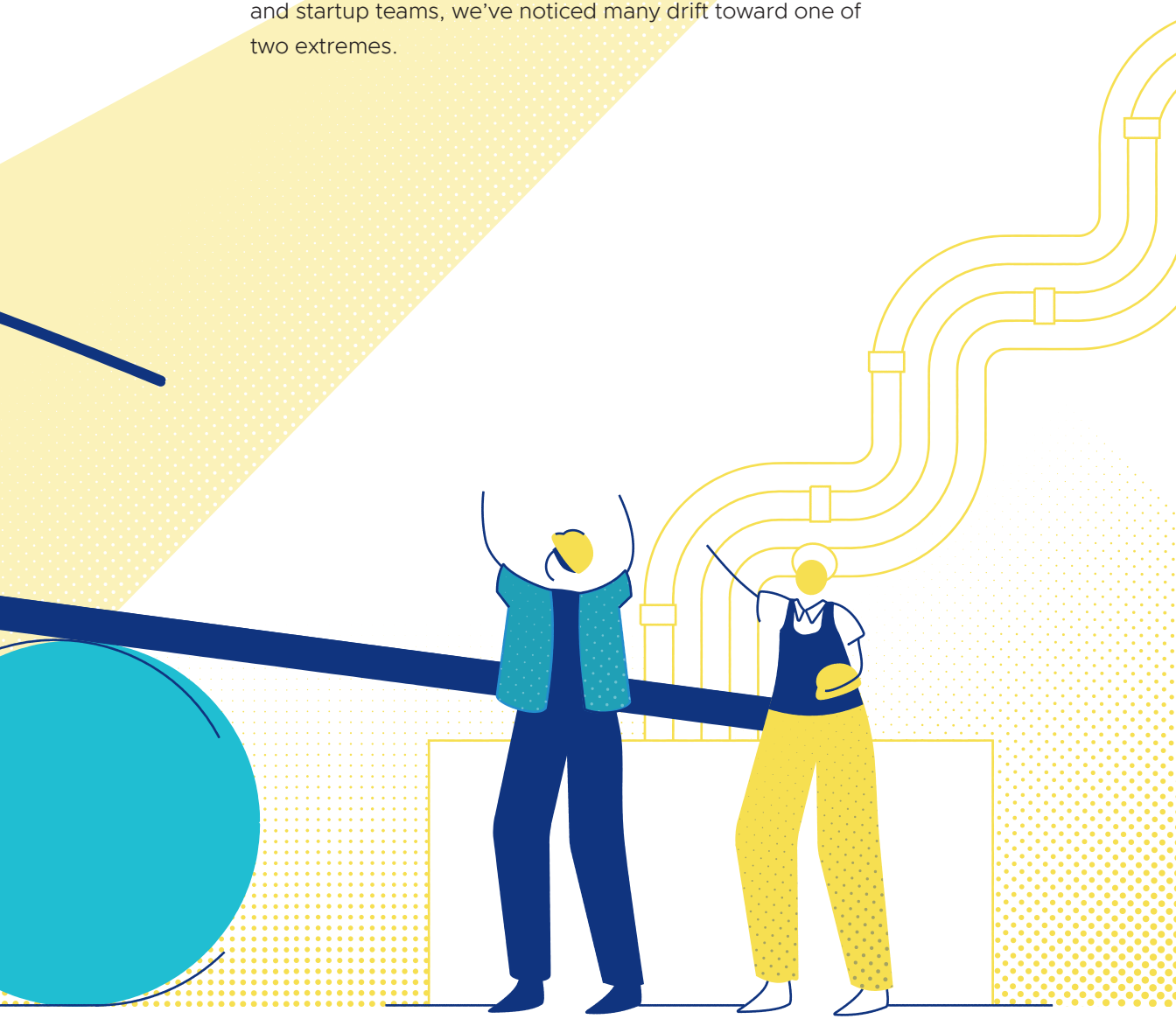
This affects enterprises even more acutely, as they commonly face these challenges:

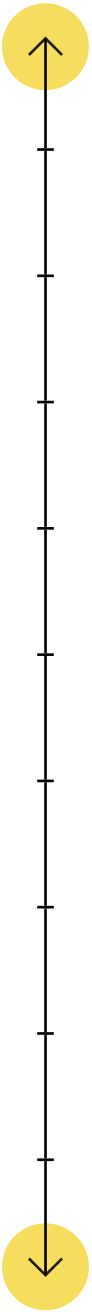
- Software products are built with a slow turnaround time.
- Products get released that don't meet customer needs.
- Competitors capitalize on opportunities with better solutions at a faster rate.

This is where design can help—it brings clarity. Design is a powerful tool that gives teams the mindset and framework necessary to make informed and strategic product decisions centered around the customer. Design helps companies respond and adapt quickly by integrating customer insights throughout the development process, ultimately connecting business drivers with software execution through customer behavior.

# Finding the Goldilocks zone

The only way to build great software products is to get working software—i.e., deployed features and functionality that people can touch and use—into the hands of users quickly so that feedback can inform product development. Balancing research with delivery doesn't always come naturally. During our work with enterprise and startup teams, we've noticed many drift toward one of two extremes.





# Too much
## research and design validation

At this extreme, teams spend months conducting research in isolation, disconnected from the realities of actual software development. Accepting nothing less than a perfect solution, teams leave no stone unturned and create massive specifications that document every minute detail of their vision. Only when this is complete do they consider handing off findings to UX or visual teams—let alone software engineers. By the time development actually starts months later, the research may be dated or the business needs have changed, rendering the specifications highly risky—"Do people still need this product?"—if not unusable.

# Not enough
## research and design validation

At the other extreme, teams view design as an aesthetic implementation step. Tasked with executing on a long list of business requirements, designers have little to no space to conduct research or validate their decisions with users. Instead they push pixels, churning out endless screens for approval and, often months later, development.

# Just enough
## research and design validation

To deliver the best solutions for users, we need to find a balance between these two extremes. That means doing enough research, at the right times, to inform iterative product development without delaying delivery of value, lengthening feedback loops, or introducing unquestioned assumptions. This approach gives a team space to focus its limited resources on exploring the most promising ideas—something that can ultimately save them time in the long run.

We think of this balance as the Goldilocks zone—not too much, not too little, but just enough research and design validation to go fast forever. And we've built our practices around it.

# The principles of Agile UCD

We deliver working software—not design artifacts—by getting our products into the hands of users as quickly as possible.

AGILE DEVELOPMENT IS DEFINED by the ability to respond to change from feedback and learnings through testing and experimentation. And you can't do that well when design is relegated to an introductory phase or only allowed to paint the product's surface. Put another way: We believe effective design is neither waterfall-style "big design up front" nor surface-level "sizzle"—it must be present throughout the product lifecycle, occurring whenever and wherever necessary.

We call this interpretation Agile UCD, and it's guided by six principles:

### Balance
Design is a balanced team sport.

### Lean
Starting with high value enables speed and reduces risk.

### Focus
Product outcomes matter more than outputs.

### Inclusivity
Good design is accessible design.

### Curiosity
Research uncovers the highest value user problems to solve.

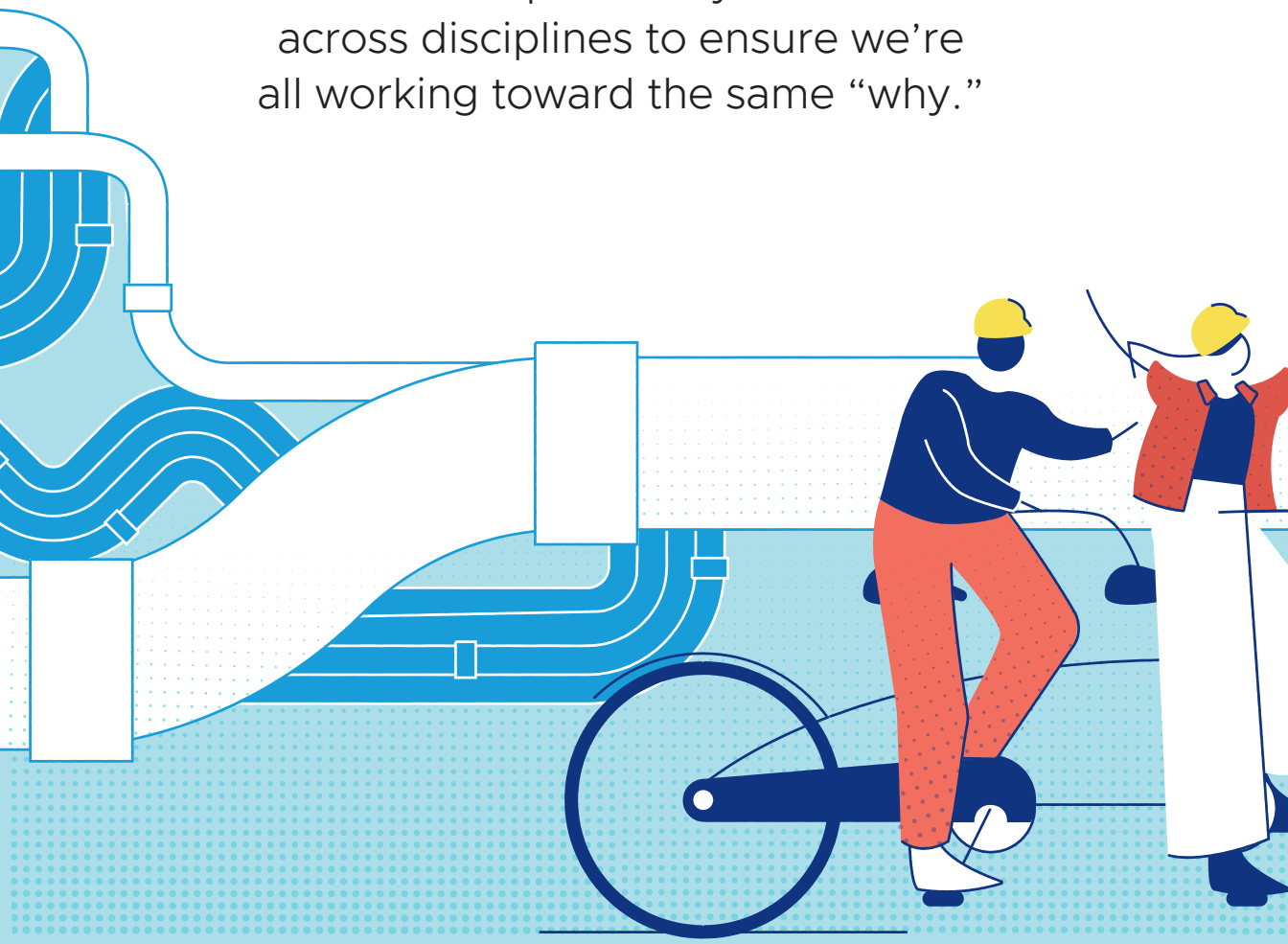
### Iteration
Internal-facing feedback fosters a culture of learning.

Principle 1

# Balance

Design is a balanced team sport

We share responsibility and context across disciplines to ensure we're all working toward the same "why."



The DNA of Tanzu Labs product teams is multidisciplinary. Each team member brings along unique expertise and diverse perspectives that everyone can leverage to make quick, informed decisions. Each person from each discipline is dedicated to one product at a time and is involved throughout the product lifecycle. This reduces feedback cycles and context sharing from hours or days to an instant.

We call this a balanced team. Most teams include at least these three roles:

- product manager
- software engineer
- product designer

The composition of the balanced team is adaptive and can be tailored to meet the unique needs of each problem space.

For instance, we'll include a data scientist on a data-heavy or machine-learning project.

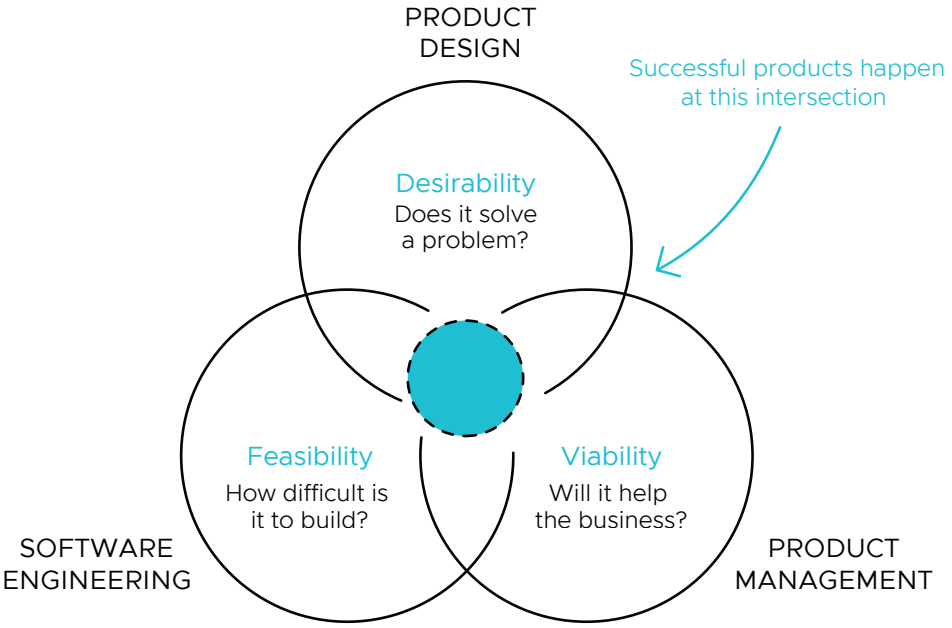
TIP — HOW TO GAUGE TEAM SIZE

## "How big should my team be?"

While we don't have a hard-and-fast rule about team size, we know that as the number of people on a team increases, so does the communication overhead. In an Agile setting, teams need to be able to make quick decisions with input from all perspectives. To that end, our product teams typically include from 4–12 people, almost always with a product manager and a designer. The number of developer pairs fluctuates depending on need.

Each role on a balanced team brings a key perspective.

A product designer practices UCD and design thinking. They're the voice of the user. Designers can derive insights from user research that enable the team to make evidence-based and strategic decisions. In this way, a product designer focuses on the experience and desirability of the product. Generalists that are fluent in many aspects of design—from research and UX to service, interaction, and visual design—allow balanced teams to move quickly, reduce handoffs, and increase shared context.



A product manager practices Lean and focuses on the viability of the product. They prioritize the team backlog ruthlessly and continuously, balancing business objectives and stakeholder vision with a deep understanding of user needs.

A software engineer practices Extreme Programming (XP) and focuses on the feasibility of the solution for the problem that the product is intended to solve. Engineers choose the technologies to use, determine how to get the product into production, and champion all technical decisions for the product.

Each person on a balanced team works toward the same set of goals:

**Business goals**
What does success look like from a business perspective? Are there any important dates to consider?

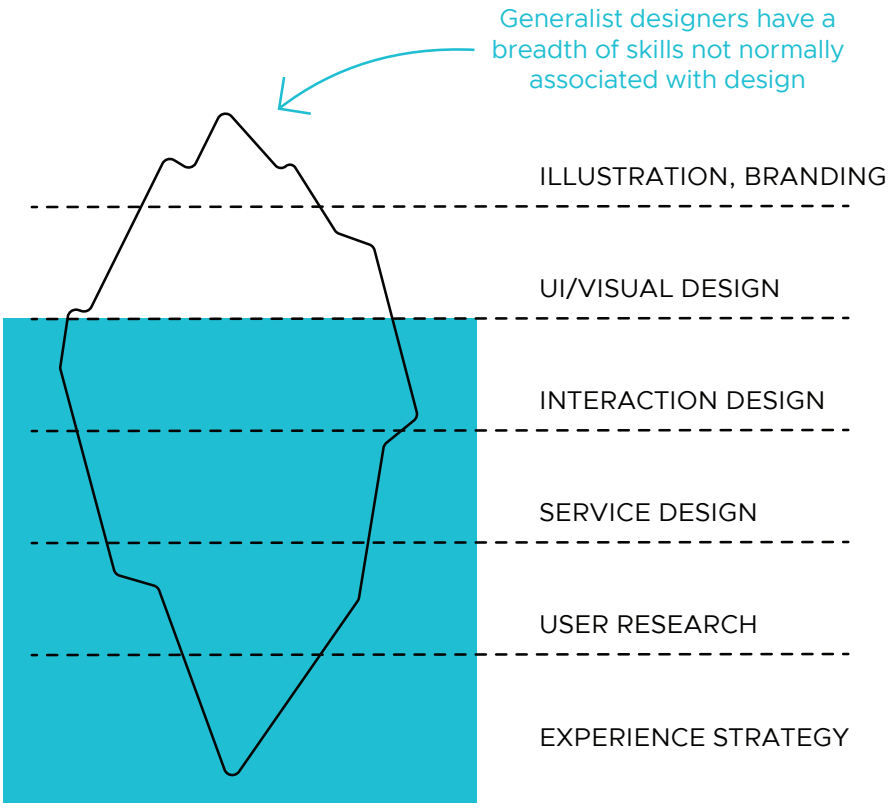
**Product goals**
What should be the impact on the end user experience?

With one team focusing on the same goals, everyone can align and have clarity. This shared understanding allows the team to move with speed and reduce waste by eliminating handoffs.

TIP — HOW TO INCORPORATE SPECIALISTS

## "What if my designers are all specialists?"

When building a team, it's important to remember that your goal is to make the product successful. If your designers are specialists, you may want to consider pairing them together—e.g., a user researcher with a UX designer—in order to increase context sharing, reduce time-consuming handoffs, and grow in each other's respective skills.


Generalist designers have a breadth of skills not normally associated with design
ILLUSTRATION, BRANDING
UI/VISUAL DESIGN
INTERACTION DESIGN
SERVICE DESIGN
USER RESEARCH
EXPERIENCE STRATEGY

## Principle 2

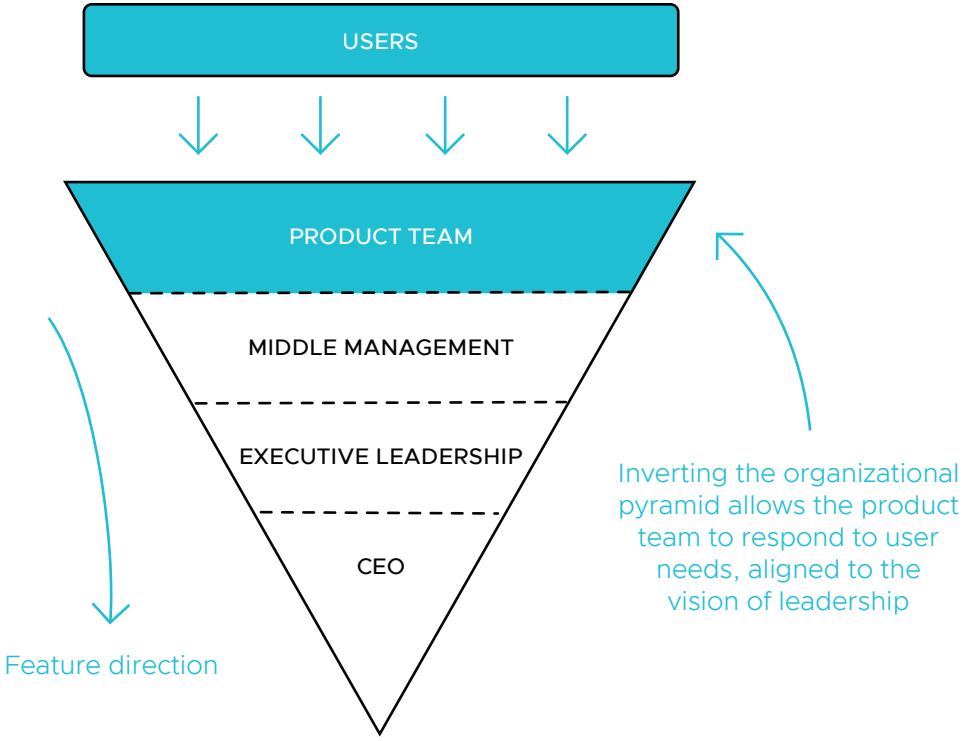
# Focus

Product outcomes matter more than outputs

We take a product mindset that prioritizes decisions driven by customer and business value, quick feedback loops, and iteration.

A team must be clear about its outcomes. An outcome is a specific, measurable statement that lets you know when you've reached your goals; it's what you expect to occur as a result of your actions. This is in contrast to defining outputs—such as a list of features to be built—without any validation or the ability to change direction if those features don't bring value to an end user.

In large organizations, defining product direction using outcomes requires that the balanced team has enough autonomy to use the expertise of the users, the business, and the technology to make product decisions that help achieve those outcomes.



## Principle 3

# Curiosity

Research uncovers the highest value user problems to solve

We understand early what a user needs to ensure we build products that solve real problems for them.

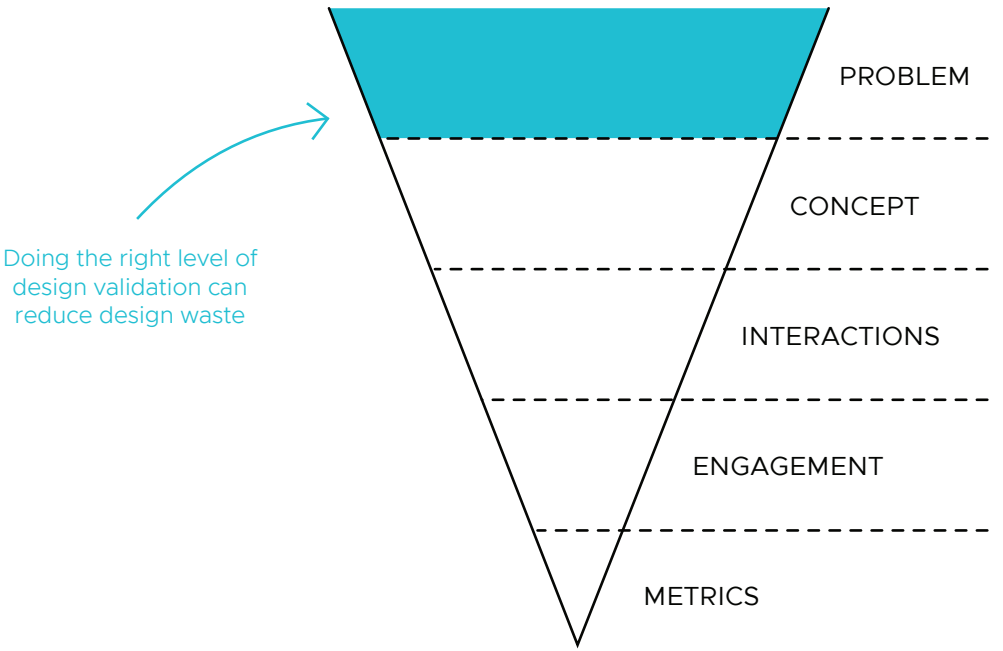
At the core of every digital product is a problem being solved for an end user. But the process of understanding which problems users are facing often gets overlooked in software development. Prematurely defining features and their functionality leads to assumptions about what the user needs are for a given problem space.

The first step in knowing what to build is empathizing with end users through a variety of research methods. Using product outcomes as our north star, we can explore a problem space and identify the best opportunities and user needs to be addressed with a software solution. We spend time learning:

- who the end user is for that product
- current user behaviors as they relate to the problem space
- problems and pain points the user is facing
- competitor products that solve a similar problem for the user

After turning research data into key insights, we map them against our business outcomes. By comparing user insights to business outcomes, we can validate that we're solving a real problem. We can also discover new opportunities that weren't apparent before.

This is just one way that design drives validation in the product development process. Our Lean UCD mindset allows us to choose the right level of validation for the context.



TIP — HOW TO CONDUCT SYSTEM RESEARCH

## “My product doesn’t have ‘end users.’ What does research look like in that case?”

Sometimes, user research isn’t about people and is instead about the interplay across services. These are critical for a system to work, but are far removed from the user. If this is the case, do what you can to understand how existing services work and what’s most important about them.

Use a machina persona to apply the user persona format to services. It’s an artifact for your team that personifies a service and helps everyone understand what it can and cannot do, which other systems it interacts with, and what its purpose is. You can make as many as necessary.

## Principle 4

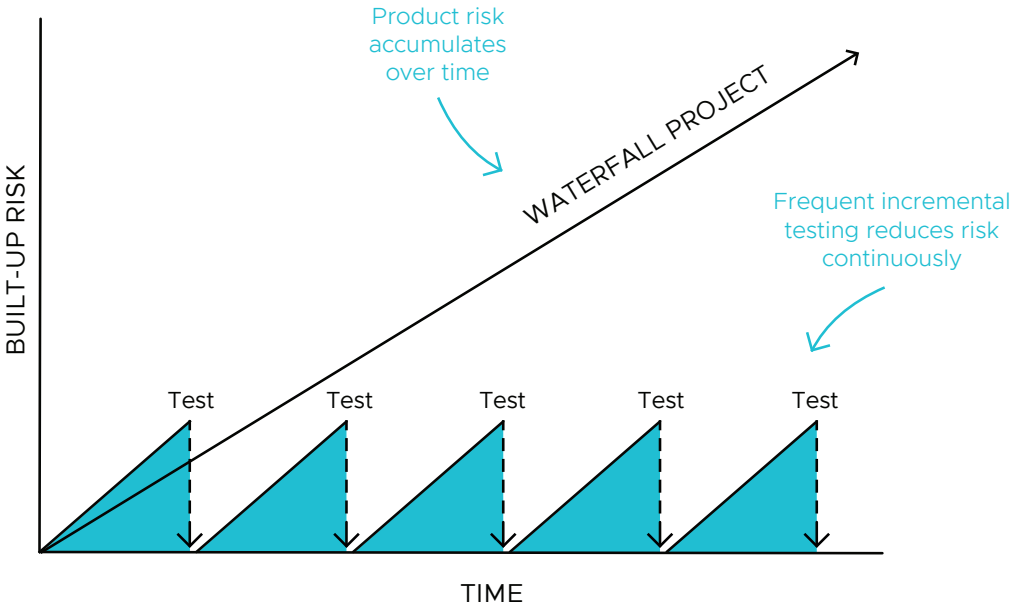
# Lean

A narrow focus enables speed and reduces risk

We build incrementally so we can test frequently and reduce the risk of building the wrong thing.

The biggest risk to any product comes from prematurely defining and building too much before it gets evaluated with real end users. Without frequent user validation, any plans for future features are just assumptions.

We see this happen often. A team will build a digital product based on extensive requirements and release it to its user base after a year or more of development, only to learn they've missed the mark: the product doesn't solve a problem for the user, is difficult to use, or both.



The best way to establish a narrow focus is to prioritize the most valuable piece of functionality that a product brings to its end users. Once that's been validated with users, we begin the iterative cycle of designing features in small increments and continuing user feedback. This tells us whether or not our product is going in the right direction, and allows us to both bring the most value to users in the shortest amount of time, and course correct before spending too much time and money building the wrong thing.

TIP — HOW TO BALANCE LEAN AND SYSTEMS THINKING

## "What if the product is part of a larger system in a complex landscape?"

Most products are connected to a larger system consisting of other products and experiences for the end user. As you start working on a new product, it's important to have a holistic understanding of the system while keeping your focus small. Before you narrow in on a thin slice, identify constraints and dependencies. This can include external teams, systems, processes, and other governing bodies.

Use a service blueprint to map out all the steps of a user's journey through a product experience. This allows the team to pinpoint where on the journey that other systems or products might connect to your users' journey. Once a service blueprint is created, it provides the big picture perspective of where a given set of features you're working on fits in.

Principle 5

# Inclusivity

Good design is accessible design

We design with diversity in mind, taking care to ensure the products we build can be used by as many people as possible.

Inclusivity and accessibility are still mys-terious topics for most people building products. It can be easy for those of us who don't often think about disability to default to a self-projecting bias, even when we may have disabilities ourselves. Personas are often riddled with bias towards ability, income, language, race, age, and gender. We may even believe common myths about digital accessibility (e.g., that it's an edge case for an insignificant subset of users or that it takes too long to be MVP-friendly). These blind spots can create a gap that often goes unchecked, leaving accessibility issues to be addressed after a product is built.

Why does this happen? Teams can find it hard to prioritize accessibility and often plan to tackle it after the product is mature—or worse, not until they get into legal trouble. But retrofitting for inclusion, while still valuable, is far more expensive and time consuming.

We design inclusively from the start because it improves the usability of our products for all our users, regardless of their ability or access needs.

Inclusive design doesn't just protect you from litigation or allow you to meet compliance requirements; being accessible from the start results in the improved quality and usability of your products every time.

And there's another reason to follow accessible design principles from the outset: return on investment. The spending power of older adults is $15 trillion USD annually[1] and the global market of people with a disability is more than 1 billion.[2]

We've seen the evidence that diverse teams build better and more inclusive products. This creates the space for the divergent perspectives that are essential to innovation. With the power to shape the future of technology, building accessibility into our design process is critical.

At Tanzu Labs, diversity and inclusion is a top priority. There's no singular approach to fostering a diverse and inclusive culture. It takes commitment, focus, and data. But our values—Do the right thing; Do what works; Be kind—have created a blueprint for our culture to create an inclusive environment that embraces iteration.

1 Paul Irving, "The Longevity Opportunity," Harvard Business Review. https://hbr.org/2018/11/the-longevity-opportunity

2 The World Bank, https://www.worldbank.org/en/topic/disability

## Principle 6

# Iteration

Internal-facing feedback fosters a culture of learning

We embrace feedback as a way to continuously learn, improve, and grow our team dynamics and organizational culture.

The technology landscape is always changing. Designers need to continuously adapt to new ways of working while building digital products in complex and high-risk environments. The model for how product designers, product managers, and software engineers work together can always be improved, and requires reflection about what's working and what isn't. This allows the design team to share knowledge and stay up to date with best practices.

At Tanzu Labs, we accomplish this through frequent, active, and useful feedback loops like Agile retrospectives. We empower practitioners to own their culture with regular internal design community events that focus on their practices. We know the design process isn't rigid and we need to provide the space for practitioners to change their approach depending on the needs of the product and the team.

We also know that it's still not the norm for many organizations to value product design as a key discipline. And even when design has been established as an organizational discipline, it often operates in a silo separately from the business stakeholders and engineers building the software. We've found that starting small and growing a design practice incrementally—e.g., by hiring a designer who's dedicated to one balanced digital product team—is the best approach for organizations that don't know where to start on their journey toward a more integrated, collaborative design structure.

TIP — HOW TO ESTABLISH A CULTURE OF LEARNING

## "How do I start implementing a culture of learning on my design team?"

Shifting culture and creating the psychological safety required for feedback won't happen overnight. A quick way to get started is to create rituals that make the space for designers to collaborate, share feedback, and exchange information. If you have teams in different cities, virtual design meetings can be a great way to foster a strong design culture that centralizes best practices.

Sometimes having a weekly design critique across teams or locations can feel like extra work and might dissuade people from participating. The Design Happy Place meeting format is a weekly ritual that brings designers together to discuss predetermined design-related topics, share new techniques, or get their work critiqued without the pressure of having to show their work every time.

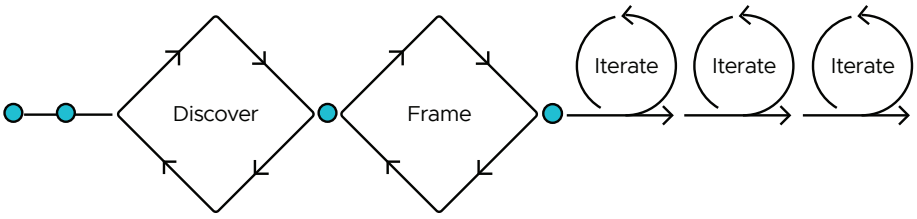
- Create a list of discussion topics with the team ahead of time to kick start conversations.
- If one person is attending remotely, then have all participants attend remotely, even if they're in the same office. This creates a more inclusive meeting culture.
- Start the meeting with the "wins and challenges" format. This encourages teammates to share things from the last week worth celebrating, as well as things they're struggling with.

# Agile UCD in practice: Discovery and Framing

We created a flexible framework for building software that solves real problems for end users—quickly.

»» THE SIX PRINCIPLES OF OUR APPROACH establish the framework for Agile UCD, but to use them in practice requires a format that's many things at once: it must be actionable, so it can get us closer to the right solution; it must be repeatable, so we can focus our energy on solving the product problems at hand; and it must be flexible, because no two product journeys are the same.

At Tanzu Labs, we call this Discovery and Framing. We'll explore this format over the next several pages, explaining many of the takeaways of each part but stopping short of offering a "how to" guide. After all, design methods are a dime a dozen these days; what we find most lacking is a focus on the "why." Once teams know "why," they can step away from prescriptive processes and instead use the right tools to build high-value software products.



# Discovery

EVERY PRODUCT JOURNEY BEGINS with Discovery. This typically happens over a fast-paced, two-week period during which we learn as much as possible about the product space, the business drivers, any existing technology or technical constraints, and the users.

During the Discovery period, teams prioritize the key problems that represent the highest business and user value so that they can use this information to inform potential solutions.

There are three necessary actions for Discovery:

1. Identify your assumptions.
2. Gather evidence and validate (or invalidate).
3. Prioritize the key problem to solve.

# Identify your assumptions

A business often begins Discovery with an opportunity or business problem they're looking to solve. This is a strong starting point. However, it's even more critical to validate that a solution has the desired outcome, is valuable to users, and is feasible to deliver.

To do this, the full balanced team works together to build a shared understanding of the business, user, and technology motivations that make up the problem space. For example, they may talk with stakeholders about their desired out-comes and the problems they believe they're solving (and for whom). The team may also conduct an exercise with a wide array of stakeholders and subject matter experts to identify any assumptions held about the problem space. These assumptions—particularly those that pose the most risk to product success—are key to guiding research efforts.

TIP — HOW TO CONDUCT SYSTEM RESEARCH

## "What if I'm already a subject matter expert in my product's space?"

Even experts stand to learn something new. While it's great to incorporate any existing evidence—as long as it's truly evidence and not conjecture—you should validate any other assumptions you have to remove the possibility of bias.

Our bias can particularly impact our assumptions about our users and their abilities, socioeconomic background, tech literacy, and language. Creating inclusive personas that challenge your assumptions about your users' abilities is a powerful way to ensure your bias is not creating a barrier for users who don't neatly align with the team's personal expectations.

- Get clear about your product intent.
- Call out the many perspectives that are represented on the team: gender, ability, language, tech literacy, etc.
- Explore alternative views and capture additional assumptions unique to your team's various perspectives.

# Gather evidence and validate (or invalidate)

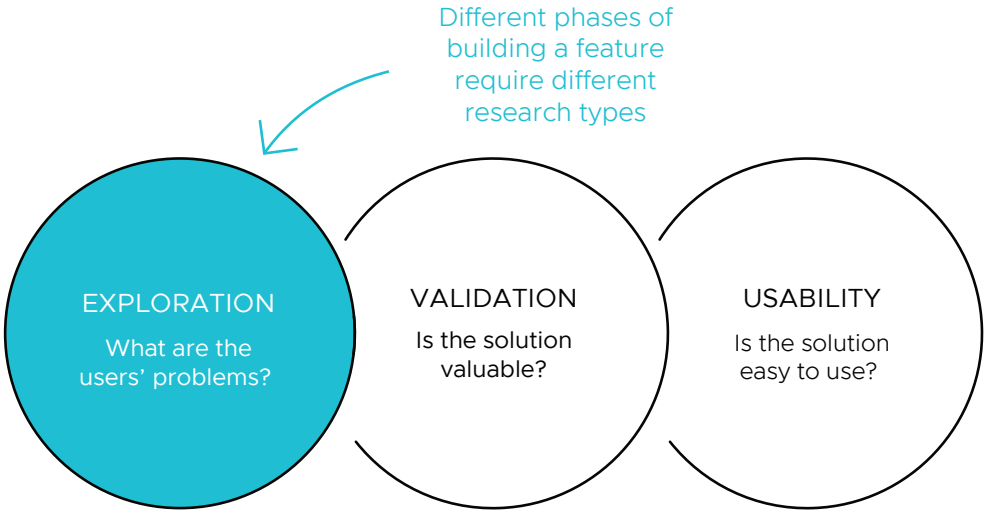
Once the underlying assumptions have been identified, we engage with end users and subject matter experts to validate or invalidate those assumptions. We use methods like 1:1 interviews with users or contextual inquiries (i.e., observing users). While these methods can often comprise much longer research efforts, our focus is specifically on validating key assumptions. This means our research can be simultaneously efficient, effective, and directional—at times requiring only a handful of users to identify trends.

At this stage, the team has likely seen and heard user comments that both confirm and conflict with their initial beliefs. To avoid confirmation bias, it's important that we step back from our personal feelings and look objectively at the notes taken in each session. By looking across all of the research data and synthesizing it into key insights, patterns, and problems, we can determine if we have sufficient evidence to validate our riskiest assumptions  and avoid narrowly focusing only on the findings that resonate with us personally.

This stage of activity is particularly exciting because it helps us discover things we didn't know that we didn't know—surprising challenges or user desires that can help us create meaningful and delightful solutions.

A word of caution: one common research pitfall is to jump into "solutioning" too quickly.

For example, we may get excited about building a possible feature or product and let that potential solution blind us to other, more interesting directions. As much as possible, we want to stay focused on needs, challenges, and behaviors during this stage.



TIP — HOW TO CONDUCT SYSTEM RESEARCH

## "What if the other disciplines can't join research?"

It's not feasible to include all team members in every round of research. To remove research silos, it's important to disseminate what's learned with the full team as frequently as possible.

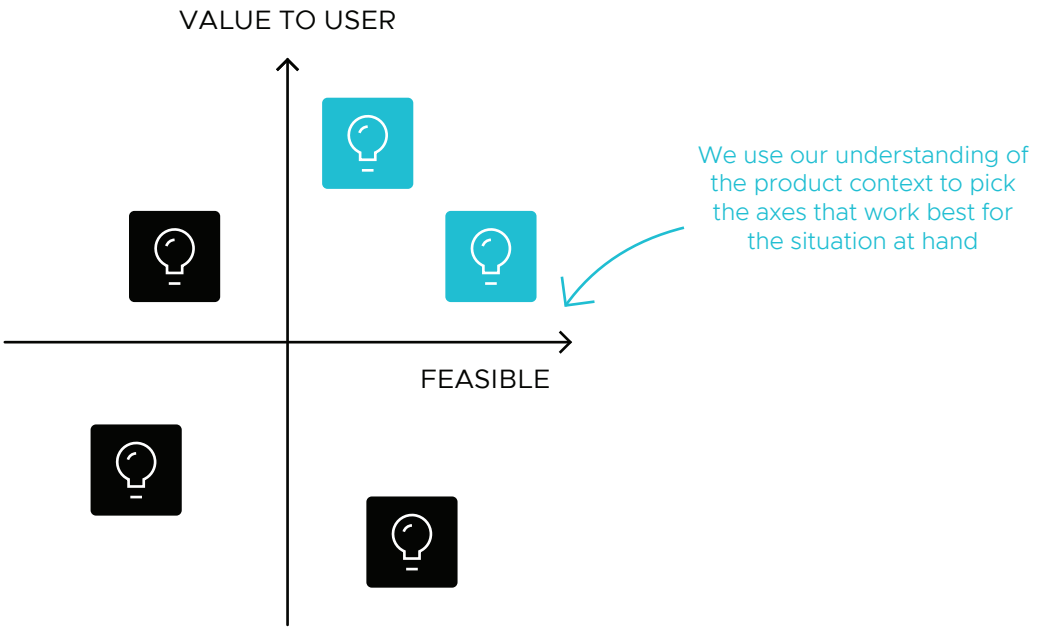
Design Coffee is a short design chat that helps all members of a team feel connected to product and design activities. Each week, for 10 minutes (coffee in hand), a team member shares one product design topic for discussion. It could be work that needs input or the rationale around a product decision. All members are invited to give input, and if necessary, make time for a longer session at another time. The designer on the team can support the team members by facilitating visually or capturing any thoughts on the whiteboard.

# Prioritize the key problem to solve

By this point, the Discovery process will have uncovered various validated user problems and invalidated myriad others. Solving any of these problems could create value for the user and the business; however, that value will only be recognized once we deliver a solution. Products often fail when they don't solve a real user problem, but they can also fail when the team making them attempts to deliver too many solutions. Prioritization is the best way to mitigate that risk.

Unfortunately, prioritization can be difficult because problems are often inter-connected and our empathy for users biases us toward taking on as much as we can. We can break through this by employing a prioritization framework, like the 2x2 matrix (on the following page) that helps balance key decision-making factors.

Ranking the problems by their relative business and user value is a great way to begin the decision-making process. A team that's shared the experience of interviewing stakeholders and users—and has synthesized those learnings—is well positioned to do the hard work of discussing this critical dimension.

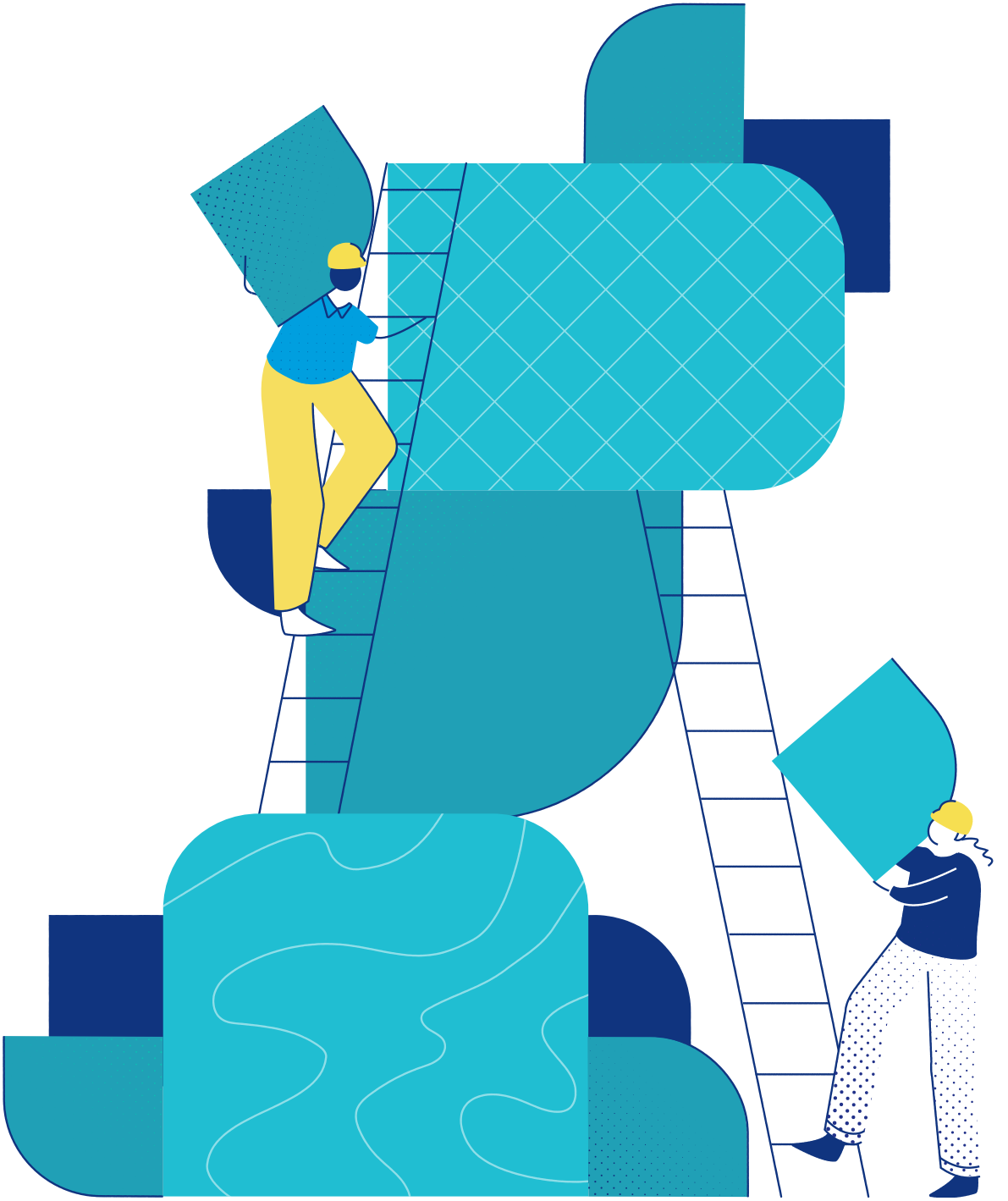

VALUE TO USER
FEASIBLE
We use our understanding of the product context to pick the axes that work best for the situation at hand

Next, we prioritize the relative feasibility or ease of solving the problems. While we won't know the full complexity of the solution at this point, we should be able to articulate whether solving the problem is more or less difficult than other problems. Having engineers as part of the team working through the Discovery process helps keep it grounded in technical expertise.

Prioritizing in this way highlights a single problem that the team has agreed is both highly valuable and feasible to deliver. Choosing to tackle this problem first lowers key risks to product success.

This ruthless prioritization of a single problem over all others can feel like a risk itself; it's important to remember that this is a starting point. This doesn't mean other problems are unimportant, or not worth solving. We're simply acknowledging that we must align the entire team around a single, tractable solution that we must deliver quickly, before we move on to the next one.

| RISK | SOLUTION |
|---|---|
| Delivering a product customers don't want/won't pay for | Start with a problem we've validated as valuable to solve and get it into the hands of customers for feedback as quickly as possible. |
| Building features that cause experience and technical bloat without delivering value | Focus on a single problem—rather than a set of interconnected problems—that the team can deliver with measurable outcomes. |
| Delaying delivery by taking on too much technical complexity too early | Choose a less complex starting point as a way to get to production early and learn as a team. |



# Framing

AFTER WE'VE PRIORITIZED the first problem to solve, we're ready to move into Framing. During this period, many potential solutions will be explored, evaluated, and iterated on. As we move through the phases, we'll continue to discover new information about our users and the technology we'll use to deliver our solution.

Framing is about answering the initial questions of solution direction so we can start building the product and get it into users' hands. We break this into three main action areas:

1. Explore many solutions.
2. Focus on a high-value starting point.
3. Run experiments to validate concepts.

# Explore many solutions

For almost any problem, there's a variety of solutions. Looking back at the insights and evidence gathered from research—particularly those that helped determine whether or not a risk assumption was validated—is important for refocusing the team.

### Going wide

Although our focus on a single, validated problem is narrow, finding the solution requires us to go wide. At this stage, we aren't trying to design the entire product, but rather generate many possible solutions that address the single prioritized problem.

Collaborative sketching exercises—like Design Studio—are one way to use the collective creativity of the team to quickly generate many options for key features. At this point, a solution may not be a software design concept. Business processes, services, technical architectures, and other approaches may provide a better solution than software, and those opportunities should be explored.

Just as we include all team members in the Discovery process to ensure a shared understanding of the problem, it's best to keep the team together during Framing. Having designers, product managers, and engineers working together to generate and evaluate solutions ensures that their unique insights and perspectives are included early in the design, and limits the number of post-design reviews and revisions.

TIP — HOW TO GET YOUR TEAM COMFORTABLE WITH SKETCHING

## "What if my team is nervous about getting involved with design?"

The designers on the team don't have a monopoly on the creative process and good ideas can come from many different places. Involve other disciplines on the team early on in the ideation process. The most important aspect of their involvement is that ideas are shared and visible to the whole team.

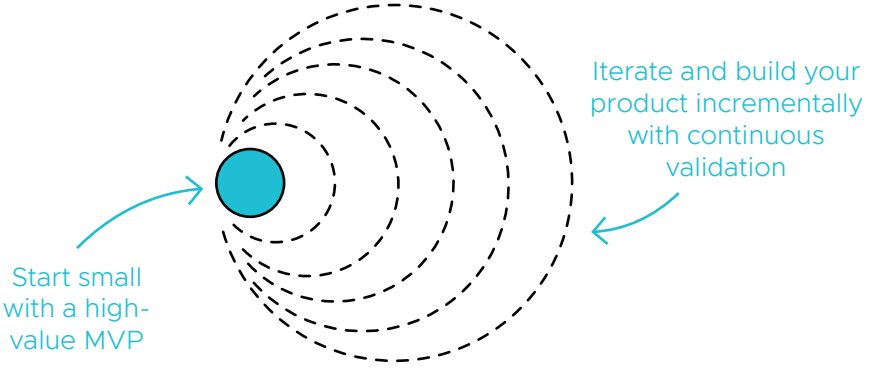
One of the easiest ways to get the team more involved in design is to have them help sketch out potential solutions in a Design Studio, typically using only Sharpies and paper. If they're nervous or think they can't draw well enough, start by demonstrating how to draw UI components, like simple boxes and lines, in order to show them that anyone can do low-fidelity sketching.

# Focus on a high-value starting point

### Prioritizing using parameters that matter

Moving quickly isn't about how fast you work—it's about how well you prioritize what you need to build and deprioritize what you don't. We do this by creating a prioritization framework for the features we've identified. We may use the same framework we used to prioritize problems: weighing user value against technical complexity, and looking for a high-value, low-complexity starting point. Other times, we may choose to tackle a more complex technical element early because it meets our criteria of reducing the most risk while delivering the most user value.

We build the highest value core functionality pared down to its simplest form in order to validate that the most important aspects of the product work with the least amount of effort. From here, we can begin evolving and iterating the product based on the initial high-value feature we've validated.



# Building in accessibility

As we start thinking about what our solution could look like, it's important to make sure we have a diversity in user feedback. Casting a wide research net can often shine a light on the ways our solutions could be excluding potential users. Getting our product in front of users with different abilities will not only help us avoid costly rework, but will also provide new insights for how users with different abilities interact with technology. This allows us to deepen our knowledge, and over time, accessibility considerations become second nature in our design process.

Being intentional about accessibility is key. Calling out accessibility goals for our design practice will act as a north star that will remind our team to watch out for any risks throughout the design process.

It's often surprising how good UX best practices—like high color contrast, simple language, and layout—can significantly reduce the need to make major changes later. These design practices benefit all users, even those without disabilities. By designing products that are more accessible for people with different abilities, the product is often made easier to use for everyone.

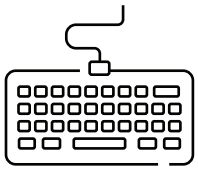
TIP — HOW TO FIND TIME FOR ACCESSIBILITY

## "What if we don't have time for accessibility?"

To reduce the cost of prioritizing accessibility, it's important to consider accessibility from the beginning of the product lifecycle when it's easiest to incorporate. There are many small steps you can take to improve your product's accessibility. From a development perspective, you can also include accessibility linters—a tool that analyzes source code to flag programming issues—or build pipeline tooling that identifies any inaccessible features.

- Make accessibility a shared team responsibility.
- Train your teams on basic accessibility guidelines and include it in all aspects of your design process (e.g., research, personas, and user stories).
- Create accessible design systems for reusable components or leverage an existing accessible design system.
- Build empathy and learn from testing with users of different abilities.
- Do early testing on your designs with resources like web browser extensions that allow you to experience your designs through the lens of users with different abilities.

# Designing for inclusion spurs innovation

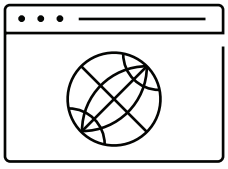


### Typewriter

Italian inventor Pellegrino Turr created the typewriter with his lover, Countess Carolina Fantoni da Fivizzano, who was beginning to lose her vision. The typewriter gave them a way to communicate—and keep those messages private—and has now evolved into the keyboard we use everyday.

### Internet

Vinton Cerf, Google's Chief Evangelist, was pivotal to the creation of the internet. He was deaf, which meant he had to share documents instead of speaking on the phone. The 1981 IP-based network grew to be the internet we can't live without today.



### SMS Texting

It's hard to imagine communicating in a world before SMS texting, but this everyday technology was the invention of Finland's Matti Makonen, who wanted to make communication easier for deaf people.

# Run experiments to test concepts

As we move through the product design process, we continually generate ideas that we believe can deliver real value for our users. As exciting as that is, these ideas contain assumptions. Some are low risk and easy to iterate on; others are high risk, and if we're wrong, we might build something of little value and waste a huge amount of effort.

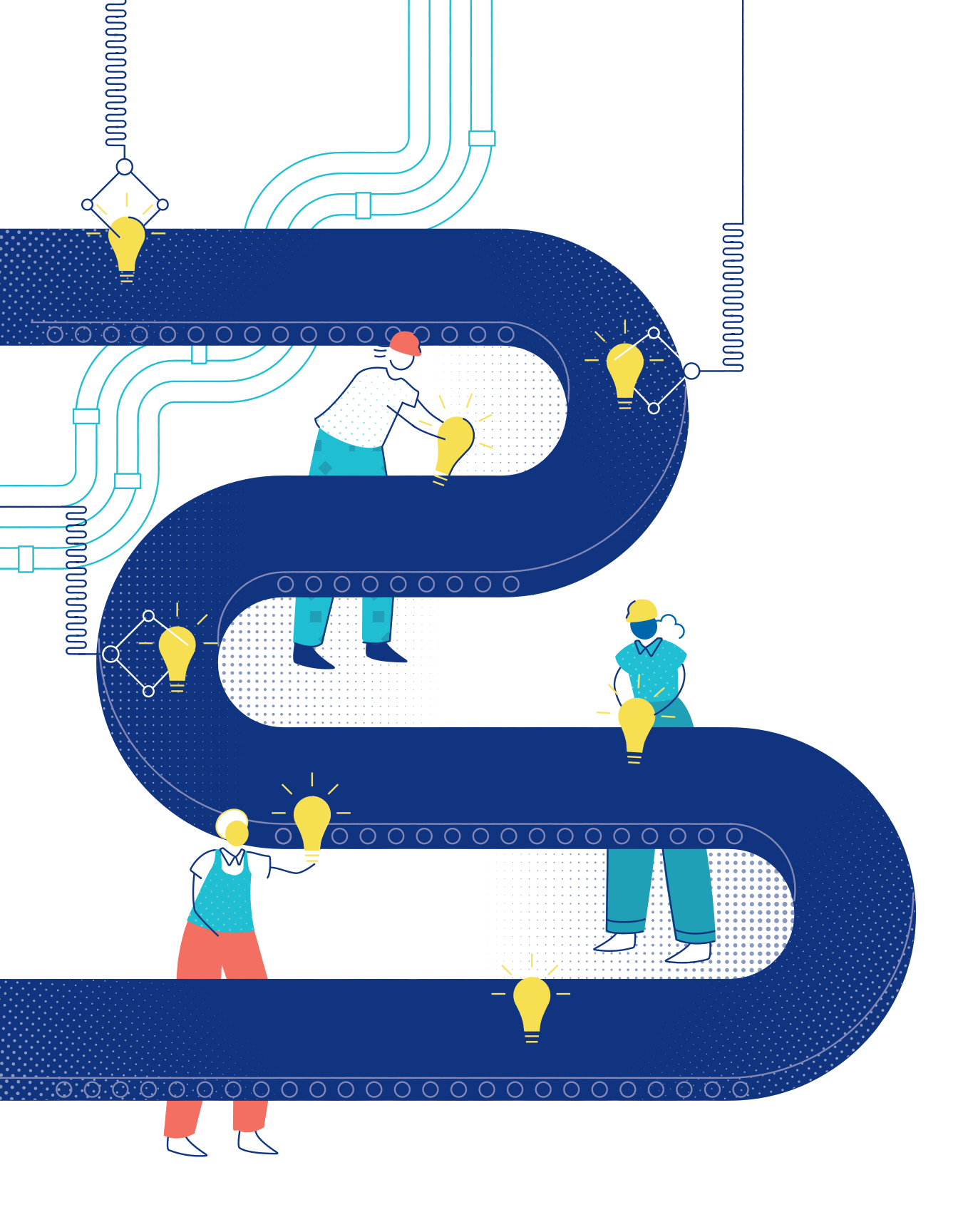
We use Lean design experiments to reduce those big risks. Lean experiments are structured, short learning activities for testing our ideas and assumptions. They set out to validate the "leap of faith" (i.e., the riskiest) assumption about a specific hypothesis in the quickest, cheapest way possible. At the end of the experiment, the team will know whether or not their solution addresses a real end user problem and will have gathered even more learnings about its users.

Lean experiments have the added benefit of reducing the cost of failure. This creates an environment of increased psychological safety for the team in which trying out new innovative ideas is welcomed and not risky.

TIP — HOW TO CONDUCT LEAN EXPERIMENTS

## "Lean experiments sound a little scary and my team is worried about failure. What should I tell them?"

Don't worry! Like scientific inquiries, Lean experiments are designed to be proven or disproven. Because of this, regardless of whether your experiment succeeds or fails, you'll still learn something helpful—you'll either feel really good about building the thing you tested or you'll avoid building something costly and time consuming.



# Continuous, incremental delivery

DISCOVERY AND FRAMING LAYS OUT the process of understanding a problem space and defining a solution to get a product or feature off the ground. The key to this process is that the initial research phase is not the only time that you're understanding user needs and defining solutions for your product.

Here's how we continue understanding user needs and validating concepts throughout the lifecycle of a product:

1. Release valuable functionality quickly.
2. Measure outcomes to determine success.
3. Grow the product through continuous validation.

# Release valuable functionality quickly

With a focused and high-value product concept coming out of Discovery and Framing, it's often tempting to continue designing new features and testing wireframes without actually releasing any software to real users. Although running Lean experiments and testing wireframes is valuable in determining initial product direction, the best test for a product is shipping it to production and using it in the real world. By narrowing focus and shipping a high-value vertical slice of the product, we've spent the minimum amount of time and money possible to find out if the core of the product works for its users.

The most common term for this idea is Minimum Viable Product (MVP). We find MVP means different things to different people, which causes confusion and uncertainty. To us, it means thinking in thin slices with end-to-end value—it's important not to consider everything all at once. Focus on a small part of the broader user journey, but one that has a beginning, middle, and end that makes sense to some-one using it. We call this a vertical slice.

Here's how we define an MVP:

### Minimum

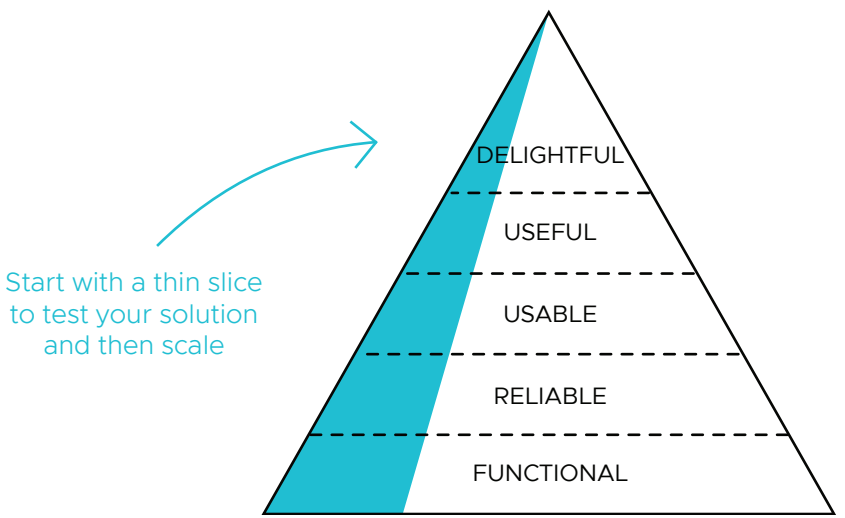
- Product contains the smallest possible feature set that delivers a valuable, end-to-end experience.
- It has only the design it absolutely needs to deliver that experience.
- It may not be immediately relevant for all users.

### Viable

- Product is of sufficient quality that it appears finished.
- It's complete enough that the user is never left hanging.
- It's been validated with users.
- It doesn't exclude based on disability or impairment.

### Product

- Product has a potential business payoff worth the investment.
- Product does not have a discrete "end," and will continue to be built upon as the team learns.



TIP — HOW TO FIND YOUR MVP

## "What if I'm rebuilding an existing product?"

An MVP is usually associated with small startups and working on brand new products, but the same principles and approaches apply in a larger enterprise setting. For different contexts, it's important to recognize that "viability" is a sliding scale that reflects the unique needs of the users in that market. The goal is not to forsake the useful, usable, and delightful aspects in pursuit of being Lean.

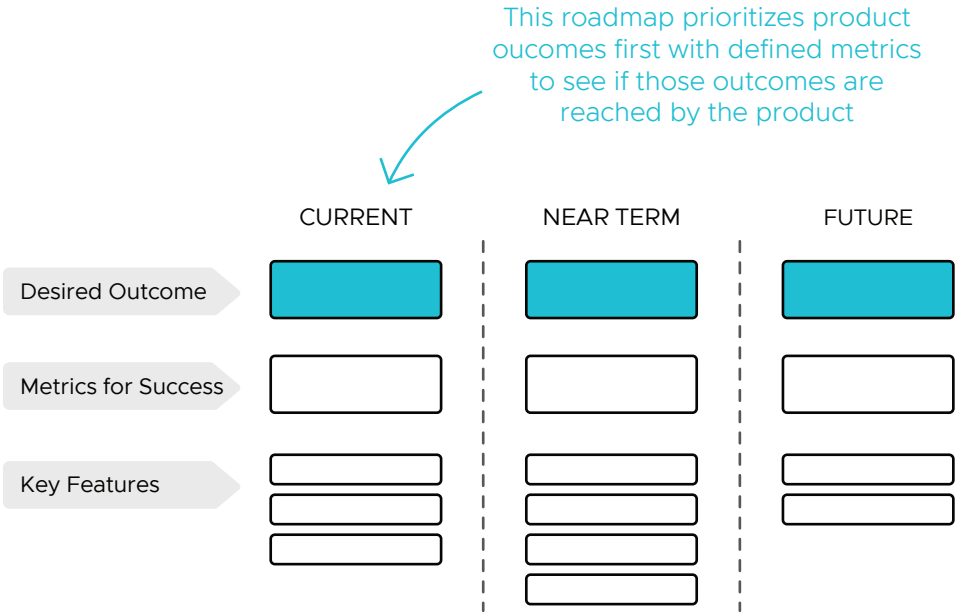
A common scenario we see involves redesigning a legacy application with a long list of features that are core to the product. Often, this means that users will continue to use the current system while the team is building a new one, starting with an MVP. When attempting to break down an existing product into smaller pieces, consider the following:

- What's the smallest standalone, end-to-end workflow that your users need to complete an important task?
- What's the most painful part of the existing experience that your tool can solve?
- What's a functionality that your users need that the current tool doesn't provide?

# Measure outcomes to determine success

The complexity of building software makes it difficult to define all of its future functionality up front. Large enterprises have been applying traditional management principles, intended for creative processes where the cost of change is extremely high.

One of the biggest challenges for product teams working in Lean and Agile is defining the future vision for a product while providing enough flexibility for the team to learn from users, test ideas, and build a product incrementally. One of the best ways to align the vision and plan for a product is to define and measure outcomes that a product aims to accomplish, allowing iterations and learning to drive product decisions toward those goals.




TIP — HOW TO BUILD AN OUTCOMES-BASED ROADMAP

## "How do I plan for the future without describing which features are needed?"

Whereas traditional roadmaps create a timeline indicating which features are expected by when, an outcomes-based roadmap tracks the outcomes that a product aims to achieve over time. These roadmaps allow a team to communicate direction, high-level strategic initiatives, and progress against those initiatives, without stipulating dates by which this progress will be achieved. This optimizes for learning and responding and recognizes the uncertainty inherent in planning for the future. What matters is not that we build feature X, but that we solve our customers' problems.

# Grow the product through continuous validation

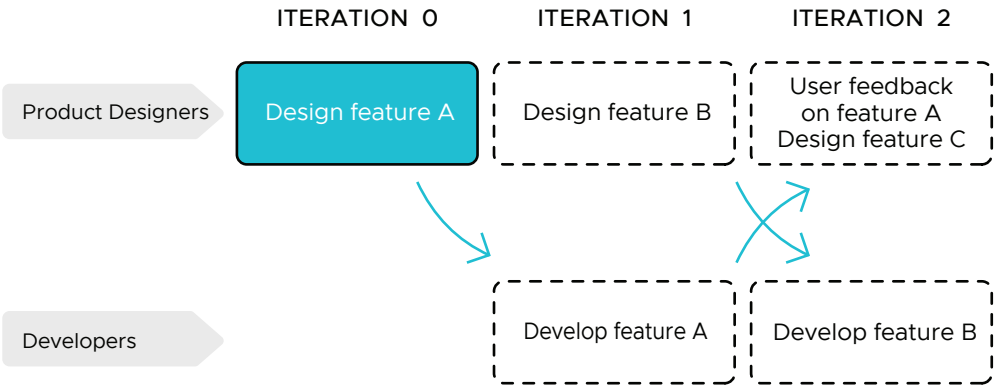
In order to build upon and iterate on an MVP, the ideas and processes described in Discovery and Framing continue as part of defining and validating new functionality for our product. Framing puts an emphasis on narrowing our scope and focusing on a vertical slice of high-value functionality. Once this functionality is released in the form of an MVP, the work begins to find out what to build next.

The team can take the following steps to determine a direction to grow our product:

- Identify the next outcome on the roadmap.
- Use feedback from the MVP to plan any necessary iterations or pivots.
- Monitor the backlog of prioritized user problems from Discovery efforts.

This is the start of what continuous validation looks like. Based on these considerations, we can make informed decisions about whether more user research is needed for the next feature, or whether the team can jump into creating solutions. Either way, once a new feature is designed, some form of validation is needed before investing time in development.

Once software engineers have begun building the MVP, product designers work a few weeks "ahead," designing new features or figuring out what to build next.




TIP — HOW TO TRACK INSIGHTS OVER TIME

## "I'm learning a lot from my users. How can I keep track of it all over time?"

It can be really hard to keep track of smaller insights week to week—especially when they aren't trends when you first learn them. You may even want to discard them, despite the fact that over time they may accumulate and become larger insights you want to act on.

One simple way to avoid having insights slip through the cracks is to create a Findings Tracker, which is usually a lightweight spreadsheet filled with outlier learning and ideas. Outliers can contain meaningful, one-off information; keeping track of them each week can give you a deeper understanding of your user's needs.

# Conclusion

Design in an Agile landscape isn't easy. It takes time and concerted effort to align the disciplines and make a space in which staying Lean, being iterative, and working toward outcomes is valued. But it can be done!

The principles outlined in this guide have become our north star since growing from the early days of design at Tanzu Labs through today. Over the years, we've iterated a lot—experimenting, failing, adjusting, and tweaking our approach—and each year we find new ways to improve. This continuous improvement has helped us grow into the design organization we are today and, more importantly, has allowed us to guide the many organizations we've worked with over the years. Now, we hope this approach can act as a guide for you on your own design journey.

Acknowledgements

This book was created by Salomé Mortazavi, Andreas Eiken, Chris Oates, Jesper Bröring, Emily Leahy, and Eric Cipra, as well as the extended team of designers at Tanzu Labs who have worked to define and iterate on our design practice. Editing by Steve Lichtenstein. Layout by Rachel Holland. Art direction by Coon Lam. Illustrations by Moe Bonneau.

VMware
Tanzu Labs

If you want to learn more about our practices or start a journey with us, visit *tanzu.vmware.com/labs*.