TA-DAA!

~~The~~ Guide to Creating a Successful Software Product

workingmouse

workingmouse

Version $\longrightarrow$ 1.0

01 December

2020

About this book

Software development is a complex industry in a state of constant flux. It can be difficult for people new to the industry to quickly understand the skills and mindset required to build a successful product.

Let’s get into it!

Contents.

YAY!

Introduction

Over the past 9 years we've built plenty of software products. For the majority of that time our mindset mirrored that of the industry – we build to our clients requirements. This meant washing our hands at the end of a project, knowing we delivered what the client wanted but not knowing whether or not the product succeeded.

On paper, it sounds fair enough. You pay someone to build a house and as long as they build the house you designed, you wouldn't expect them to check in and see if you like it...

But as an industry and a company, we felt we had to do better. Rather than build solutions blindly, we're shifting the focus to building successful products.

Why does product planning and measurement matter?

It's easy to look back at successful products and think, 'of course that worked, how could it not?' Of course Uber worked, people were fed up with the dominance of the taxi industry. Or, of course Xero succeeded, people were favouring the cloud for its flexibility. These observations are rarely as obvious at the time these products were started. The mindset also neglects the thousands of micro-decisions that had a big impact on their success. A big reason why we wanted to create this book for our audience is to unpack on a more granular level the activities, decisions and measurements that can help create a successful software product.

While some of the examples used in this book are household names (Uber, Xero, Airbnb), that does not set the definition of 'success' as a multi-billion dollar application. These learnings are just as important for a business implementing software that streamlines their internal workflow as they are for a startup. Every product has the potential to succeed or fail – regardless of its scale.

We can spend the next ten pages outlining all the reasons why product planning and measurement is important. But we'll keep it more succinct.

Proper planning allows you to build a unified, succinct solution. We've seen products in the past suffer when planning wasn't prioritised and a number of pivots were required during the build stage. Planning is not isolated to the product itself. It's just as important to plan and align your business strategy with your product.

Measurements on the other hand are your pulse checks. Proper planning will create a hypothesis. But only by releasing and measuring the impact that a product has can we determine whether that hypothesis was successful. Doing this poorly may lead to making decisions based on only a segment of your user base. Failing to measure altogether means there's a lack of insight into how the product is performing and where the opportunities are to improve engagement.

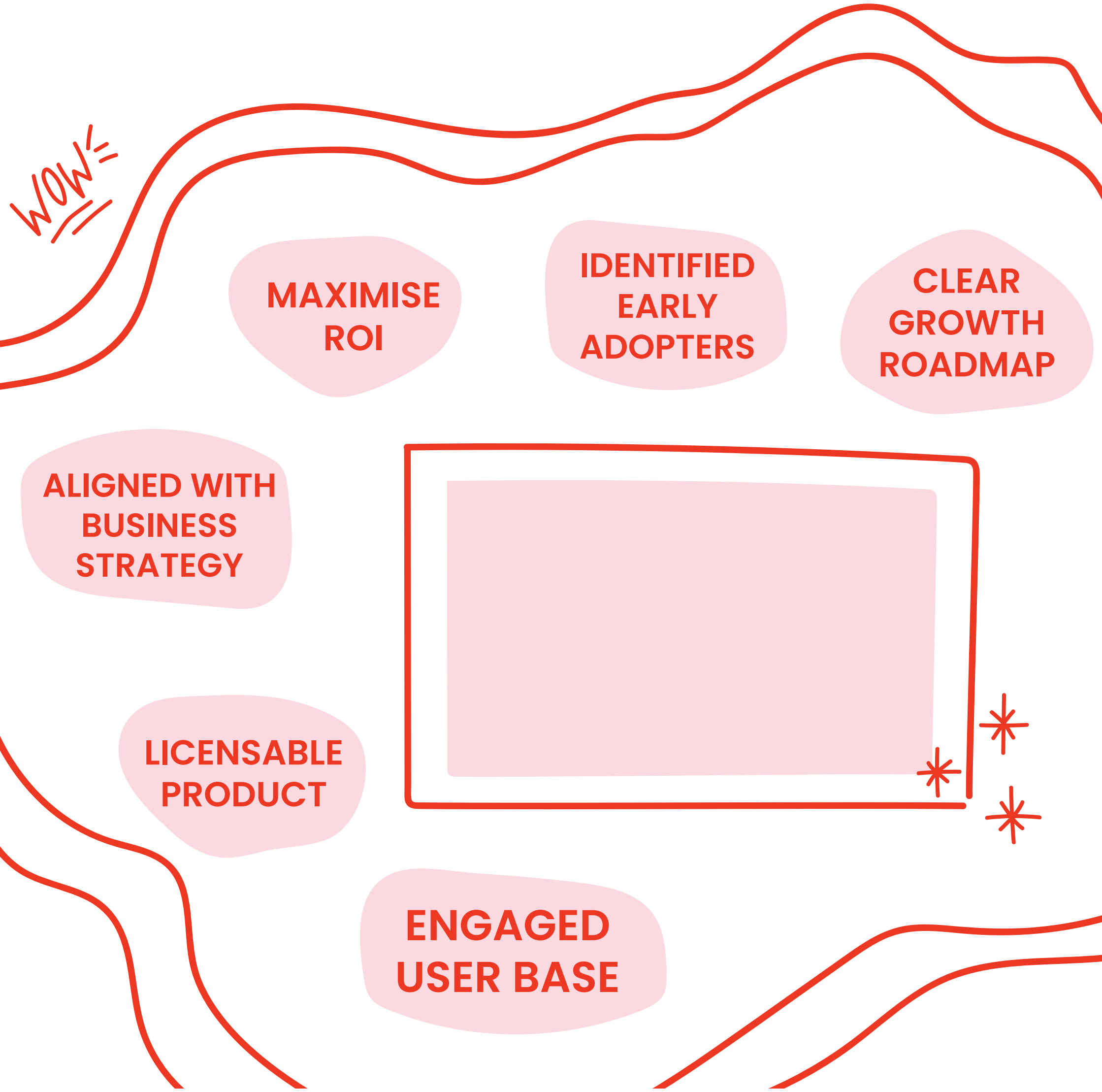
How it impacts your business

There are a wide range of reasons as to why a business might develop a software product. Some of the more common themes we’ve seen are;

- To help scale up
- Improve engagement with customers
- Reduce licensing costs
- Create a new revenue stream, and;
- Create a point of difference over competitors.

While the reasons for developing software may be varied and the products themselves differ, the framework for maximising your chance to succeed remains consistent. That is the purpose of this book – to communicate and demonstrate the framework that can help your business create a successful product.

Without limiting the outcomes of a successful product, some of the benefits are illustrated in the image below.



"

Until you've built a great product, almost nothing else matters.

Sam Altman

Starting with a problem

Start at the start, not the end.

This is an area that can trap a lot of products. So often we have visions of the shiny, finished product in our head and race to get there as soon as possible. We frequently see businesses come to us with a complete requirements list, asking 'can you build this?'

The issue is they can't answer the most fundamental question; what problem is this looking to solve? Or, there are multiple answers to that same question. Without a unified problem statement understood by all stakeholders, there can be a big gap in expectations. This doesn't just pose a risk from an internal (client) and external (development team) perspective. A product can reach so many areas of a business there are normally multiple stakeholders involved. The challenge is to ensure that each stakeholder is satisfied with the problem statement and clearly understands how it relates to their function. The problem statement also gives us a north star to keep everyone moving in the right direction.

What didn’t work?

To fully comprehend the value delivered in this process it is important to firstly understand the old way of building software. The old process involved capturing what we define as a list of ‘Epics’. These are large volumes of work that could then be broken down and prioritised into smaller ‘User Stories’ during a designated scoping period. As mentioned above, this encouraged the list of requirements based approach.

It often amounted to an unvalidated wish list based upon a product owner’s vision. We found that designing a solution based on this approach meant that too many assumptions were made without any validation.

The issues that came from this approach were:

- Projects failing to truly capture customers needs and provide a viable solution to their end-users.
- Differing expectations between the requirements list and the scoped backlog.
- Building based on unvalidated assumptions
- Larger scoping periods which ultimately takes longer and creates bigger minimum viable products (more on this shortly).
- Didn’t properly leverage the skills and capabilities of the development team.

Activity time

Time required: 2 hours

Team needed: All stakeholders (includes internal and external development team)

Desired outputs:

- Develop the success criteria
- Problem statement and
- A contextual understanding of why the project is being undertaken.

Method:

1. Map out the current user journey without your custom software or other products available.
2. List all the problems you have to solve.
3. Prioritise these problems, from most pressing to least.
4. Ask why. Why do these problems exist? This will allow you to document any assumptions you've made.
5. Align on a single problem statement to tackle first. This should be framed around the highest priority problem you're looking to solve.
6. Iterate. If the first problem statement doesn't satisfy all stakeholders then iterate until the best statement emerges.

Adopting an MVP mindset

An MVP is the smallest version of your product with enough features to satisfy your initial customer base. It goes by many different names; minimum viable product, minimum valuable product, most valuable player. Fine, that last one was a not so subtle basketball reference. Regardless of the name, the purpose of an MVP is to gain insights into your user's behaviour without having to fully develop your product.

The sooner you can get your product to market, the sooner you can start gathering feedback direct from market. We want to avoid the nightmare where millions of dollars are invested in creating a polished, final version only to realise people aren't using it the way you thought they would.

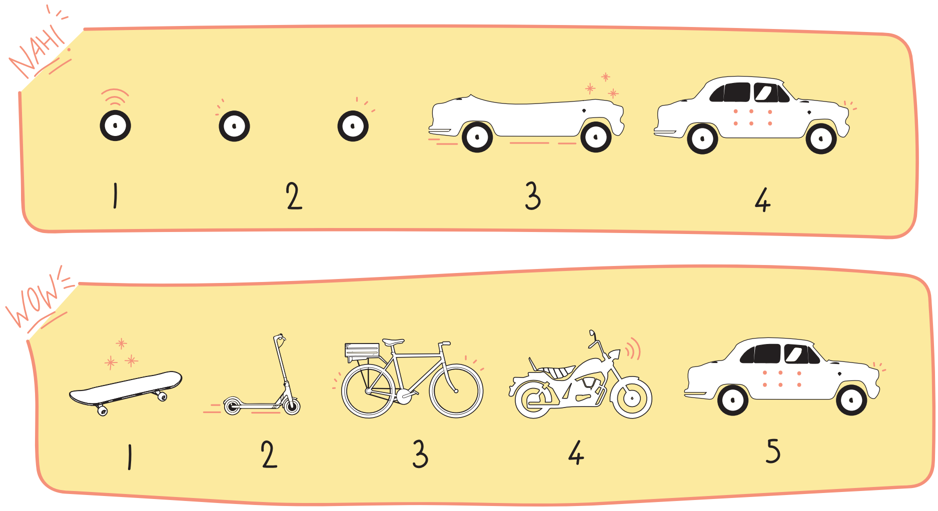
The story of Clipchamp is a great example of why an MVP mindset can be critical in finding success. Founded by a group of former SAP employees in 2013, the company was originally called TranscodeCentral and started out as an attempt to build a distributed supercomputer using people's browsers to speed up the transcoding of videos (translating video from one code to another) – an action that's critical for online streaming.

The company launched, failed to attract any users. The team developed a video compression and conversion tool as a small part of this project, and while the supercomputer project didn't work out, the video compression tool was gaining popularity.

The company rebranded to Clipchamp and recognised that once their first version was launched, they had to adapt and change in order to succeed. What made the initial tool so popular was that users didn't have to download or install anything, "you could just record a video in the browser" and it worked on any operating system.

Now the company has amassed a user base of over 6.5 million people around the globe, including employees from major corporations such as Google, Tesla, Microsoft, Deloitte and BHP. While user research can help mitigate some of the product/market fit risks, it isn't as effective as releasing an MVP and making live learnings.

Usable > likeable > lovable
The image below is a great way to conceptualise how to build a product.



The car example is quite popular and has been used to illustrate the MVP mindset.

If we take the top roadmap as an example, it's probably likely that the brief was 'to build a car.' The development team then went through the stages of building parts of the car until it was finally assembled. Until the final version is finished, you have an unhappy product owner since the car can't be used.

Now let's look at the bottom roadmap. Instead of asking for a car, the brief was centered around a problem statement; 'I want to be able to travel faster.' At every release the product can be used and tested, even if it doesn't yet match the final vision. Sure, the product owner may not be happy with commuting around using a skateboard but it's infinitely more useful than a wheel.

It all comes back to learning early and often. Ask yourself what is the cheapest and fastest way you can start learning?

Finding your customer centric focus

Starting with a problem is a fantastic mindset to have. However to take the next step, it helps to add a framework to that mindset. There are several tools that aim to do this, but our personal favourite is the lean UX canvas.

Lean UX Canvas
It was first articulated by Jeff Gothelf and has undergone a couple of version changes. In summary Jeff developed the canvas for the following purposes:

- A customer-centric cross team facilitation tool.
- Help the team focus on 'Why' they are doing the work.
- A strong foundation for teams to adopt agile.
- Ensure learning takes place every iteration.
- To expose gaps in a team's understanding.
- A first step to shift the conversation from outputs to outcomes.

We found that the original canvas was a great starting point – but ultimately made a few tweaks to better match it to the software development industry. The process is actually straightforward and following it yourself or with your own team is a great framework for finding a customer centric problem statement.

"

The only way to win is to learn faster than anyone else.

The Lean Startup

Lean UX Canvas

Project name:

Business Problem
What problem does the business have that you are trying to solve?

Hint: Consider your current offerings and how they deliver value, changes in the market, delivery channels, competitive threats and customer behaviour.

Solutions
What can we make that will solve our business problem and meet the needs of our customers at the same time?
List product, feature, or enhancement ideas here.

Business outcomes
How will you know you solved the business problem? What will you measure?

Hint: What will people/users be doing differently if your solutions work? Consider metrics that indicate customer success like average order value, time on site, and retention rate.

Users
What types (i.e. personas) of users and customers should you focus on first?

Hint: Who buys your product or service? Who Uses it? Who configures it? Etc

User outcomes & Benefits
Why would your users seek out your product or service? What benefit would they gain from using it? What behaviour change can we observe that tells us they've achieved their goals?

Hint: Save money, get a promotion, spend more time with family.

Hypotheses
Combine the assumptions from 2, 3, 4 & 5 into the following hypothesis statement:
"We believe that (business outcome) will be achieved is (user) attains (benfit) with (feature)."

Hint: Each hypothesis should focus on one feature only.

What's the most important thing we need to learn first?
For each hypothesis from Box 6, identify its riskiest assumptions. Then determine the riskiest one right now. This is the assumption the will cause the entire idea to fall if it's wrong.

Hint: In the early stages of a hypothesis focus on risks to value rather than feasibility.

What's the least amount of work we need to do, to learn the most important thing?
Design experiments to learn as fast as you can whether your riskiest assumption is true or false.

Focus of the MVP

Problem Statement	Top 3 Business Outcomes	Top 3 User Outcomes

Lean UX Canvas

Project name:

Business Problem	**Solutions**	**Business outcomes**
Users		**User outcomes & Benefits**
Hypotheses	**What's the most important thing we need to learn first?**	**What's the least amount of work we need to do, to learn the most important thing?**

Focus of the MVP

Problem Statement	**Top 3 Business Outcomes**	**Top 3 User Outcomes**

Activity time

Complete the Lean UX Canvas

Time Required: 30 mins – 1 hour
Team needed: All stakeholders

Method:

Progress through the Canvas.
Here is more information on each of the steps:

1. Business problem: What problem does the business have that you're trying to solve?
2. Business outcomes: How will you know you solved the business problem? What will you measure? (Hint: What will your users be doing differently if your solutions work? Consider metrics that indicate customer success like average order value, time on site and retention rate)
3. Users: What types (personas) of users and customers should you focus on first? (Hint: Who buys the product or service? Who uses it? Who configures it?)
4. User outcomes & benefits: Why would your users seek out your product? What benefit would they gain from using it? What behaviour change can you observe that tells you they've achieved their goal?
5. Solutions: What can you make that will solve your business problem and meet the needs of your customers at the same time? List product, feature or enhancement ideas here.

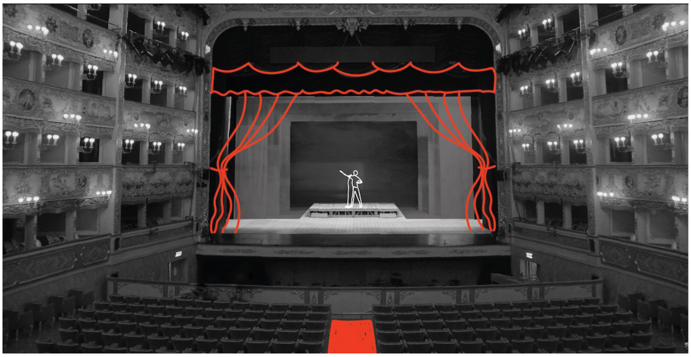
6. Hypotheses: List out any assumptions this project is making in order to succeed.
7. What's the most important thing you need to learn first: Identify the riskiest assumption/hypothesis that existsright now. It may be something that will cause the product to fail if it's wrong. (Hint: Focus on the risks to value, rather than feasibility).
8. What's the least amount of work you can do to learn the most important thing?: Brainstorm experiments to validate whether your riskiest assumption is true of false.

Focus of the MVP

1. Problem statement: Using the activity completed earlier, populate your problem statement.
2. Top 3 business outcomes: Select the top 3 business outcomes to focus on for the MVP. These metrics will be measured to evaluate the success of the product.
3. Top 3 user outcomes: Select the top 3 user outcomes to focus on for the MVP. These metrics will be measured to evaluate the success of the product.

Using the Lean UX Canvas to align all stakeholders on the project and create a single, customer centric focus will help you set your project up for success.

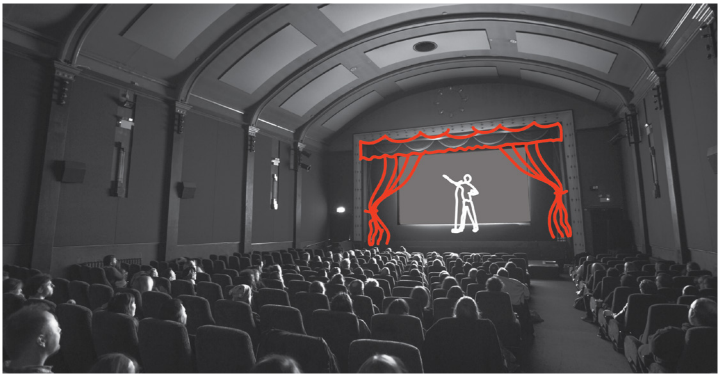
The importance of user research



It was the 1920s. The RKO theatre stood tall and dazzling in the neighbourhood of Flushing, Queens. It was schmick with its popular movies, travelling shows, a glamorous interior with a sleek red carpet that made you feel like every night was a special occasion. But no one came.

Families and farmers would walk on by without a second glance. The theatre was falling fast and so the owners decided to bring on an industrial designer to help boost interest.
His name was Henry Dreyfuss.
When Dreyfuss came on board, he did what he thought the people wanted. He slashed movie prices, tripled the showings and gave away free food. None of it worked.

Burdened with this problem he paced the halls looking for answers till he landed in the theatre lobby. Here, he overheard someone express how afraid they were of messing up such rich carpet with their muddy shoes.
And then it clicked.
The neighbourhood was home to a lot of workmen who would have found the appearance of the plush red carpet to actually be rather intimidating.



Within the next day Dreyfuss had the red carpet replaced with plain rubber matting and one by one, people came in. Eventually crowds filled the theatre and RKO became a historic landmark where it still stands today.

In the theatres case, the problem wasn't that they were lacking in their offerings. The problem was that the theatre was seen as "too nice" for its location, and by extension, its customers. When we initiate the conversation and create an open dialogue, we begin to understand more about what it is people actually need. This often requires us to look past the sometimes-obvious solution and dig deep to articulate how our users think and feel. Sometimes they might not even realise that was the problem in the first place. Just like the red carpet.

Integrating this process also assists with saving time and saving money. Two valuable and indispensable resources when building a new product. When we spend time designing the right product first, the rest comes much simpler and reduces the risk of having to continuously pivot.

How to conduct user interviews

Evidently, user research is important. The best way to conduct user research at such an early stage in product development is through interviews. A discovery interview provides a first hand account on why someone chooses to use your product or service and what tasks they seek to accomplish through it.

These sessions provide a crisper image on how you might position (or re-position) your offering. These are typically carried out in the early stages of the design process, but can really be performed when you need a pulse check.
We'll outline the process of running user interviews for your product. Remember, it's just as important to conduct user interviews for internal software (ie. users within your business) as it is for external users.

Step 1: Validate the method.
This is a simple first step but it acts as a quick sanity check; are user interviews the best way of learning what I need to know?

Step 2: Establish the objective.
Write a list of things you want to learn from the interviews. Limiting the number of objectives to 3 - 5 is effective for keeping the plan direct and concise. You can always start wide, brainstorm a list and prioritise.
Tip: Write your objectives with action verbs and ensure there is a measurable outcome. Try to avoid 'learn' or 'understand' as they can be more difficult to evaluate.

Step 3: Recruit
Recruit the people who represent your target group best. Define

a list of characteristics or criteria to ensure the people you recruit can provide relevant insights. If you have multiple target groups, recruit evenly from both.

There are a few different ways to find the right people.

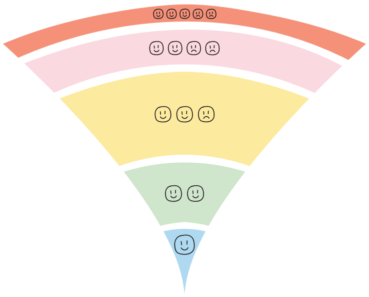
- Your user base. If you already have a user base this is a great place to start. Where better to find people that represent your user group than your users themselves.
- Agencies. There are people who specialise in connecting you with the right participants. E.g. usertesting.com or askable.com
- Social media. Powerful for expanding your reach - throw up a post on your LinkedIn, Instagram or Facebook.
- The street. If you're still looking for participants, try hitting the streets.

How many people do you recruit?

As a general rule of thumb, start off with 5 participants and scale up if you need to.

Norman Nielsen and Tom Landauer conducted research on usability tests and found that after sitting with the third participant, they had already observed much of the same things from the first and second user.

"As you add more and more users, you learn less and less"



Step 4: Prepare a guide
The general structure we follow for discovery interviews is:

- Introduction
- Background
- Activities
- Wrap up

There is no uniform way to conduct the activities as it really depends on your study. Try out one of these:
'Thinking about when…'

This is great for gaining insights on your participants personal experience. This activity asks the participant to reflect on a specific moment and recount what happened. It helps with answering those questions discussed earlier, such as how and when someone interacts with your offering.

Mapping the journey
Similar to 'Thinking about when…', you could also ask the participant to walk you through their process. Build a map of based on what they did, how they felt and what they were thinking at each stage. Combining these three factors provides you with a richer picture of their journey.

Card sorting
This activity is useful for gauging your participants preferences. It asks them to order a list of statements based on importance or relevance and getting them to talk about why they ranked them that way. This is a great hybrid for generating that quantitative and qualitative data.

Setting a growth strategy

Building a high quality product is only one piece to the puzzle. Success is often measured through the number of users, the valuation and the growth of a product. After all, why build something if no one's going to use it?

That is why it's critical to align your product with your growth strategy. There are a number of different variations and lines of thought when it comes to growth strategies. We're not a sales company and we won't pretend to be thought leaders in the area. But it's important to give thought to the strategy and feed that back into the product.

Sales driven growth

Traditionally, many businesses succeeded through a sales driven growth strategy. Create a product that people needed and would pay for and let the sales team do the rest. Once a sale is made, the job is done. A sales driven approach relies heavily on outbound sales, personal relationships, one-to-one meetings, customised sales processes, and historically longer cycles. Think back to IBM in the late 90's, early 00's.

There is still plenty of merit in adopting a sales driven strategy. In some industries, it's pivotal. If you do adopt a sales strategy then it is necessary to focus on metrics like number of sales, average sales cycle time and customer acquisition costs.

If your product is aimed at consumers or even small to medium sized businesses then a sales driven strategy may not be best suited. Because of the high-touch nature of one to one meetings and a custom sales process, the cost per sale can be quite significant. In order to justify that cost, you want a significant return on investment – a return that may not be feasible for SME's and consumers. That's why IBM, SAP and Oracle focus their efforts on enterprise sales. It may take 6-18 months to secure a sale, but that is justified by the millions of dollars in return.

Changing tides

Post GFC 2008/2009 saw software as a service (SaaS) starting to really have an impact. Bessemer popularised 5 metrics that cloud companies should monitor. Because SaaS was a fairly new model a lot of work was done in the VC space to determine the value of a SaaS company. Out of this came the Pirate metrics, the Rule of 40 and a number of others.

Many to understand the balance of growth and profit to make sure both these normally opposing forces are balanced. In 2010 most companies had adopted a sales driven strategy. At that time the theory mentioned holding back sales hires before hitting KPI's and the changing the roles of sales teams – for example hunters vs farmers. The last decade has seen the rise of Product Led Growth (PLG) - a move away from the traditional sales driven strategy.

Product led growth

What is it?
From the outside, product led growth can seem a bit like magic.

Imagine you're desperate for a video conferencing solution after buying a puppy and working from home for a few weeks.
So, you install Zoom and tell your team about it.
Your team installs it and realise that they love it.
Word spreads throughout the office – all of a sudden Zoom is the video conferencing tool of choice within the business.

From Zoom's perspective, they haven't had to invest any time or resources into making the sale. They might not even know how it happened. But because the product was so good, it spread far beyond its initial reach.
Atlassian's S-1 says, "We recognize that users drive the adoption and proliferation of our products."
Product led growth is a strategy that is fuelled by user interaction with the product and is designed to drive rapid expansion as a company scales.

Does it work?
Developed and championed by the likes of Atlassian, it is now an essential part of the playbook of companies like Zoom, Slack and numerous other successful SaaS companies.
Product led businesses are performing well. The public companies that have embraced product led growth beat their peers across nearly every SaaS value driver. They're growing faster, demonstrate better margins and trade at a 50% premium relative to forward revenue. The median public PLG company is worth 2x that of the broader SaaS index ($6.8B vs. $3.4B).
On the back of this success, sales led SaaS companies are now being pushed to adopt product led growth to reinvigorate their growth.

It's worth taking a look at the growth rates of SaaS companies that have adopted product led growth versus those that haven't. It stands to reason that at an early stage, when the user count is low, having a quality product won't necessarily have the same level of impact.

Top Quartile Growth Rates by ARR Scale

	PLG Companies	Non-PLG Companies
<$1M ARR	116%	148%
$1-2.5M ARR	185%	150%
$2.5-10M ARR	91%	105%
$10-20M ARR	88%	72%
$20-50M ARR	65%	54%
>$50M ARR	50%	30%

Image 1: *Growth rates for product companies vs sales driven companies*

Even within a product led growth strategy there are different paths a business can take.

Approach 1: Manage product interactions

If using traditional sales and marketing, you can engage and guide customers through the product. Once you've successfully learned which interactions are needed to facilitate a high quality user experience you can automate or build that into the product. This approach does create more flexibility at that early experimental stage.

Approach 2: Avoid sales and marketing
The other option is to take a hard-line stance. By avoiding any sales and marketing from the start, you can instead focus on building the best possible product. This can be a hard approach to persist with, especially when you're struggling with growing your user base and revenue.

So then, sales or product?
I wish we could unequivocally say which approach would work for you. Unfortunately there's no right answer and it may in fact be a combination of both.

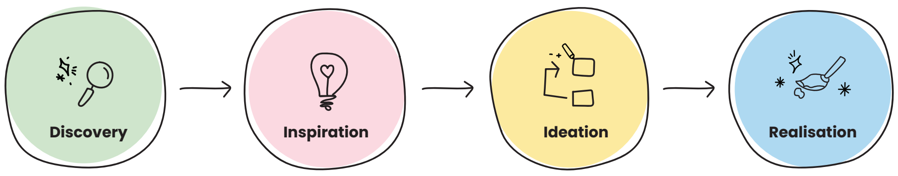
Kyle Poyar
VP Marketing Strategy at Openview states the following:
There is still room for sales in a product led business, but sales should start to look more like customer success. It's particularly urgent that all customer-facing employees become fluent in the product. Moreover, the sales team should prioritise their time based on an account's activity in the product, leveraging a product qualified lead (PQL) methodology rather than just marketing qualification.

All of these initiatives require a rock-solid analytics foundation. You'll want to capture product usage activity across features and user cohorts. Product data should then be connected with other systems of engagement (e.g. CRM or marketing automation). Most importantly, make sure that key employees have self-service access to this data in order to make data-driven decisions.
One thing we can say is that a product led growth strategy is tough to execute. The majority of times it comes down to having a well-defined problem that is applicable to a motivated user base.

Finding your solution

We’ve just outlined how important it is to start with a problem. One of the core reasons is that it sets up the process of finding a solution. Following these steps stops you from falling for the most common mistake when building software - jumping straight to a requirements list.

Now that you have a well-documented problem statement, you’re probably wondering how that is transformed into a solution. At WorkingMouse, we use a four phase process to find the best possible solution.


Discovery
Inspiration
Ideation
Realisation

Shouldn’t it be easy?

After completing the lean UX canvas, conducting a number of discovery interviews and setting a growth strategy you might have a clear idea of how the product should work. But how can you be sure you’re building the best solution if you haven’t considered any others? How can you be sure you got the best deal on that washing machine if you only went to the one store? Let’s use an example problem statement.

How might we connect farmers with consumers to minimise distribution costs?

It’s a pretty bold problem statement that might make us unpopular with some food retailers.

Solution 1 - The Farmers New Market
An online marketplace for farmers to sell and consumers to buy fresh produce locally.

Solution 2 - Micro-farm: small on waste, big on taste
A micro-brewery model which encourages more restaurants to grow and sell their produce in urban areas.

Solution 3 - The Flying Farmer
Drone delivery from farm to door
Commercially, it may be unlikely that these solutions would succeed. If they did, I’d be retired living on an exotic beach rather than writing this guide. It does however illustrate the point that given a problem statement, there is rarely a single solution.

We mentioned earlier that there are four phases to finding a solution. Let's explore these in more depth.

Discovery

This is where the team and the stakeholders pin down exactly who the users are and what they really need for the project. It is a natural flow on from discovering the problem statement.

Inspiration

The inspiration phase is similar to the activity conducted above; throwing ideas around until something sticks. Usually it helps to see how ideas look visually so we'd recommend using basic sketches during the inspiration phase.

Ideation

The ideation phase consists of refining your product's concept and MVP. We recommend using wireframes to help bring the idea to life. In creating the wireframes, you should get a clearer idea of the functionality that needs to be built (which will in turn form the backlog).

Realisation

The final phase is to realise the solution. To do this, there are a number of artefacts you or your development team should create. These include; a high-fidelity prototype, a high-level architecture of the product, a detailed backlog with estimations and acceptance criteria. Once you have the high-fidelity prototype, that's the perfect opportunity to get early feedback on your solution from your users.

At the end of these four phases you should have a perfect blueprint to build your product. Don't underestimate the importance of a well documented backlog and prototype. There can be misaligned expectations when interpreting only written or only visual artefacts. The combination of both will ensure you and your development company are on the same page.

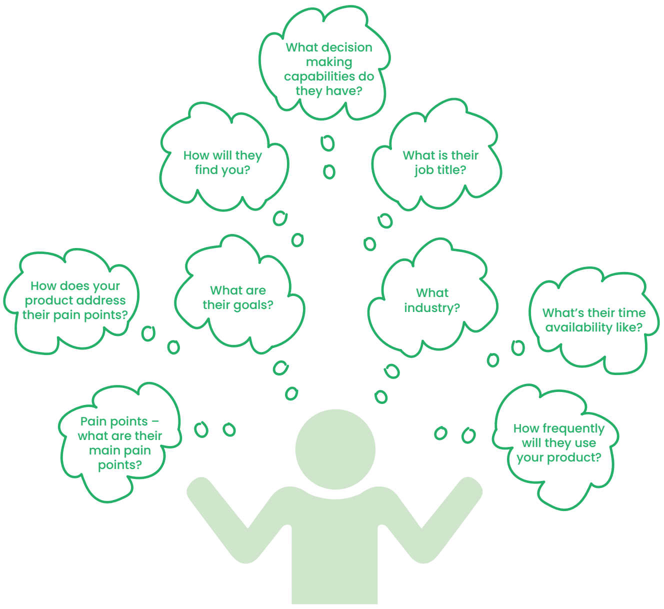
It's more than likely that you'll rely on the development company to produce the design artefacts. After all, that is what they specialise in. There are still areas to focus your attention as a product owner. We'll go into more detail about how to validate your assumptions and the process of building metrics into the design of the product.

Identifying your user/buyer personas

This should be one of the easiest questions to answer. You started with a framework of solving a problem, so it should be as simple as answering who you solved that problem for. However it's unlikely that this is just one type of user.

For example, you might solve a problem that helps cross functional teams collaborate. In that case; you could have designers, developers, business analysts and project managers all collaborating on your product. Each user persona has its own likes, dislikes, goals and problems. This is important to understand not only from a product development perspective but also from a marketing and sales perspective.

We've found that it helps to put a framework against each persona. Feel free to use the framework below or adapt it to meet your needs.


What decision making capabilities do they have?
How will they find you?
What is their job title?
How does your product address their pain points?
What are their goals?
What industry?
What's their time availability like?
Pain points – what are their main pain points?
How frequently will they use your product?

These personas should drive everything you do. How does adding a new feature impact persona X or Y? Even WorkingMouse as a services business has buyer personas that we live and breathe.

How many buyer personas should you have?
This is a difficult question to answer. Some products may have a single persona while others may have 10-20. The more you have, the more difficult it becomes to satisfy them all. We'd recommend trying to cap it at 5.

Building in metrics

This is perhaps one of the most important sections of this guide. If you take nothing else from this, understand the importance of building in metrics and analysing the performance of your product.

The metrics that you build into your product should be linked; firstly, to the success criteria you identified when completing the lean UX canvas and secondly, to any areas or assumptions that are not yet validated. You might be tempted to measure everything and we certainly won't discourage that. Having a detailed understanding of how your product is used isn't a bad thing. But depending on the size of the product, that can sometimes get burdensome to review. Instead try to pick 5-8 key metrics you want to focus on.

The Silicon Valley mindset is to put these statistics front and centre. For example, use a TV at your office as a live display of the data. This helps to get your team bought in and striving for the same results.

How to measure the success of your product is a tough question to answer. It generally depends on the type of goal you’re trying to measure. Some goals may be quantitative, which are easily measured by analytics tools. Qualitative goals are harder to measure.

Qualitative vs quantitative measurements
A quantitative goal is anything that can be measured through metrics. For example, how many users signed up this week? They are relatively straightforward and are easier to measure and draw insights from over time. We would always recommend trying to set quantitative goals.
However there are times where this may not be possible or you may want longer responses. For example, how did adding a feature improve the user experience? Now, we could get users to quantify their experience before and after the feature on a 1 – 10 scale. This would mean we miss out on potentially valuable subjective responses.

“I didn’t understand some elements of the reporting feature but I found others really valuable.” If you’re only measuring quantitative goals then this user may have kept their experience level the same which can be misleading as there were parts of the feature they liked as well as parts they disliked. That’s why it’s important to keep a mix of quantitative and qualitative goals. We find that creating an environment for qualitative measurements during user testing and research works well alongside regularly tracked quantitative data. While the key format for qualitative measurement will likely be a survey or interview, it gets a bit more complex for tools that measure quantitative data. We’ve outlined a few of the tools we recommend below.

Tools to consider

Smartlook
Smartlook is great at recording event based measurements. An event might be a user pressing a button that requests a consultation, or completion of a payment. They are important actions that you want visibility across. Smartlook also has a recording function available if you want a more detailed look into how users are behaving.

Google Analytics & Firebase
These two are pretty well known. We're a fan of them for those more general usage analytics. For example, return visitors vs new visitors, and average user session data.

Hotjar
Hotjar has a great heat map feature. A heat map is a static picture of a page with click data overlaid. It shows which call to actions have successfully attracted your users' interests. By manipulating the order or design of call to actions, you can use heat maps to evaluate its success.

CRM Integration
Integrating an application into an existing CRM like Pipedrive or Hubspot can align the product to business objectives. For example, if the product facilitates B2B sales, but those conversions happen outside of the application itself, CRM integration can help create visibility.

How they can be implemented

All analytics tools require some form of integration. Some are as easy as inserting a tracking snippet into the codebase while others require more comprehensive integrations. It tends to depend on the complexity of the data manipulation/capture and the sophistication of the analytics tool you're using.

It's worthwhile keeping this in mind during the design and development of your product. While we always facilitate the data and analytics conversation with our clients, some development companies may not. In this case, you may have to raise it before the backlog is documented.

Why measure?

Perhaps the biggest benefit that comes from measuring product performance is the ability to make data-driven decisions. That way, we completely remove the risk of making inaccurate assumptions.

These decisions become critical when iterating on the first version of your product and adding new features. They can show which features are needed most, or the impact that a new feature has on engagement. We like to think that every step taken is forwards, but the data helps prove (or disprove) that.

Build your solution

While it may be tempting to take a backseat to the development team during the build stage there is still plenty of work to be done. This is especially true for agile projects. As outlined earlier, this resource does not intend to outline the way software products are developed. There is already a great resource called the Way of Working that covers this.

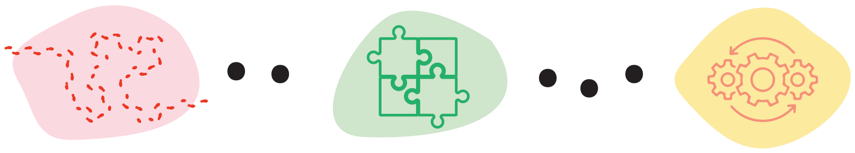
Ideally, the development agency you have engaged provides guidance on the responsibilities of a product owner during development. We believe clients should place a strong focus on;

- Managing the backlog using agile feedback loops,
- Mastering user acceptance testing,
- Getting on top of scope creep.

These responsibilities remain with the product owner and directly impact the success of the application.

Using agile feedback loops

Feedback loops are key to manoeuvring through development effectively. They are used to refine the first version (or MVP) that is released to users. We believe there are three stages any successful product will progress through in its lifetime.



Stage 1
Walkthrough

Stage 2
Systemise

Stage 3
Optimise

Your end goal should be for any user to pick up your app and instantly know what to do, or the onboarding is clear enough that they can learn everything they need to without contacting support. However this isn't where you'll start. Initially you'll likely need to walk users through the application, instructing them where they can find certain features and what those features do. Because you're diligently validating all assumptions, you've likely already conducted a few walkthrough's during the 'finding your solution' stage. It's still a good idea to conduct more walkthrough interviews during development, especially in the earlier iterations.

Because you've walked through the product with users, you know what confuses them and what's preventing them from picking it up and knowing what to do. The next step is to use this initial feedback to systemise. Verbal communication is not scalable, a systemised product is.

The final (and in all likelihood, the longest) stage is to optimise the product. The beauty of feedback loops is that you're able to continuously refine the product by listening to users and incorporating those learnings into later iterations. We'll dive deeper into how to make those learnings using analytics in the 'release and grow' chapter below.

Mastering Testing

While the primary responsibility for a product owner is managing the backlog (this will be discussed further below), the secondary responsibility is testing.

Testing comes in two forms and for two purposes. Firstly, there is user testing. This is similar to the theory we discussed when looking at user interviews. The only difference is that instead of using a prototype to test the usability, you're using the product itself. Because it isn't a basic prototype, you can get more ambitious with setting a challenge. For example you may instruct the user to log an action and then find the report that relates to it.

This guide is centred around an agile project, so there's no need to wait until the end of development to conduct user testing. Equally, I would be hesitant to recommend it after the very first iteration. The sweet spot lies somewhere in between and may depend on your comfort level with the progress of your product. I won't go into more detail on user testing, there should be plenty of takeaways from the user interviews section. The second part of testing that you need to be aware of is user acceptance testing.

The process of conducting user acceptance testing (or UAT's) may vary depending on the software development company you decide to engage. However it's an important area to understand, especially when the responsibility resides with you as the product owner.

Why do we need user acceptance tests?
The purpose of a user acceptance test is to verify the functionality created by the development team matches the requirement in the backlog. This is mitigated through acceptance criteria before the ticket is built, but re-confirmed through a user acceptance test.

It's important to acknowledge that when building a commercial product, you need multiple levels of quality control. The first 2-4 levels generally reside with the development company. However, it should always be elevated back up to you as the product owner and client.

Consider a barber shop. You sit down and tell them what you want. They take those directions and cut your hair. Throughout the haircut they'll ask "how's this length?" or "is this what you were thinking?" That's the barber industry's version of user acceptance testing. Without asking those questions, you might be walking out with a pretty average haircut you'll regret for the next few weeks. Similarly, without diligently performing user acceptance tests when building a software product, you may end up with a product you regret building.

How do they work?
A UAT is performing an action and checking the result. If I click on this button, is a new contact created? There may be multiple results you're checking for depending on the complexity of the requirement. That's where you'll likely be relying on the software development company's expertise to extract acceptance criteria. That criteria forms the base for what you're testing against.

The main takeaway here is that user acceptance testing should be prioritised. It is an opportunity to confirm that the requirement is functional and matches your expectation.

Getting on top of scope creep

Earlier we discussed the importance of finding a solution before building it. This process is called scoping. Scope creep happens when a project diverges from the discovered solution without documentation, risk management or discussion.

When a product's requirements grow unchecked, this increases development time. In turn, this increases the cost of implementation and muddies the waters of what is to be delivered and why. It can also have the effect of leading to a disjointed final product. Adding features in a piecemeal fashion can fail to take into consideration the product as a whole. Put simply: too many feature requests can derail a project.

The slippery slope

During development product owners and other stakeholders can often get very enthusiastic about the power of the software. This fires up the imagination and can lead to new ideas or additions to the product that weren't prioritised during the scope.

Project managers naturally want to ensure the client is happy with the outcome of development, so the first impulse to these change requests is to simply accept them and add them onto the existing stack of work. This is a mistake, regardless of how big the ticket is. Any change to the implementation of the product will affect the backlog, time, cost and final deliverable.

Too much scope creep will lead to a turbulent project and have a detrimental effect on the product itself. We've seen our fair share of projects that needed to be rescued because another development company allowed the scope to creep, resulting in an over-budget, half-finished product.

How to avoid it

Be a ruthless product owner

- It’s normal to have multiple stakeholders involved in a project. Everyone will have their own opinion on how the product should work. That’s why it’s important to have a single product owner who gets to make the final call on what features are developed. As a product owner you need to be ruthless when it comes to scope creep. Don’t take new feature development lightly!

Track changes

- Most experienced development companies should already be doing this. The backlog is the ordered list of everything that is needed in a given product and is the source of truth for the development team. We provide an estimate against the backlog before development starts so product owners can get a scientific breakdown of the time and costs to create the product. Without tracking changes, the available budget your business has set aside to develop the product may be exhausted before must-have features are built.

Review estimates and re-prioritise

- If the development company you have engaged or your internal developers don’t provide an estimate, ask for one. Scope creep will impact development time. It’s in your best interest to know exactly what that impact is. Having the estimate against the feature will allow you to determine whether it’s worth extending the time (and cost) or prioritising above other features.

TA-DAA! Release and grow

If you make it this far, well done. It's important to celebrate milestones, regardless of whether you go down the road of an official launch with the works (press release, marketing and sales push etc) or a more low-key, staged release.

Keep in mind that no product is ever 'finished.' Software will always need to be modernised to leverage the most recent and innovative technologies. This is why you need a development company (or internal developers) to support the product while it grows and scales. As you've adopted an MVP mindset, there is likely to be a roadmap of product development still on your radar. Keep this in mind but be prepared to be flexible as you make learnings from user data.

The product owners time after releasing tends to be split in three directions – providing first level support for users on the application, co-ordinating bug fixes/new feature development with developers and analysing user data. We won't go into any depth on the first two responsibilities, that's covered in other documentation (the Way of Working). Instead, we will focus on analysing data and evaluating the position of your product.

Synthesising your data

As we outlined in 2.3 Building in metrics, it's so important to capture data about how your application is used. But, capturing the data is only the first step. Next, you'll have to synthesise it in order to make smart decisions. In this regard, the best tactic you can learn is how to block out the noise.

These are the strategies we recommend to block out the noise and synthesise your data.

1. Look for themes

Don't make decisions based on a single user's feedback. Everyone has an opinion but very rarely are you building a product for user A. The obvious exception is software that is developed to be used by a small number of enterprises. See if you can find themes in quantitative and qualitative feedback. Be sure not to dive straight into building a solution, it's still important to follow the steps of problem – validate – solve – build.

2. Selective hearing
User/buyer personas are different user groups that you have identified a problem for and the software will solve. These are the people you concentrate your marketing and sales efforts on. Unfortunately, not every user will fit within your persona list. Some may be using the product out of curiousity, others may have been recommended it without actually having the problem you solved. Regardless, it's important to cut through the noise of all the feedback and listen to those that sit within a user/buyer persona first.

3. Don't ask people what features they want
This is one of the harder steps to follow. It goes back to the mindset of starting with a problem. Rather than asking what features should be built, instead try asking about how you can improve their experience with your product. It might uncover a usability or navigation problem where you previously didn't know there was one.

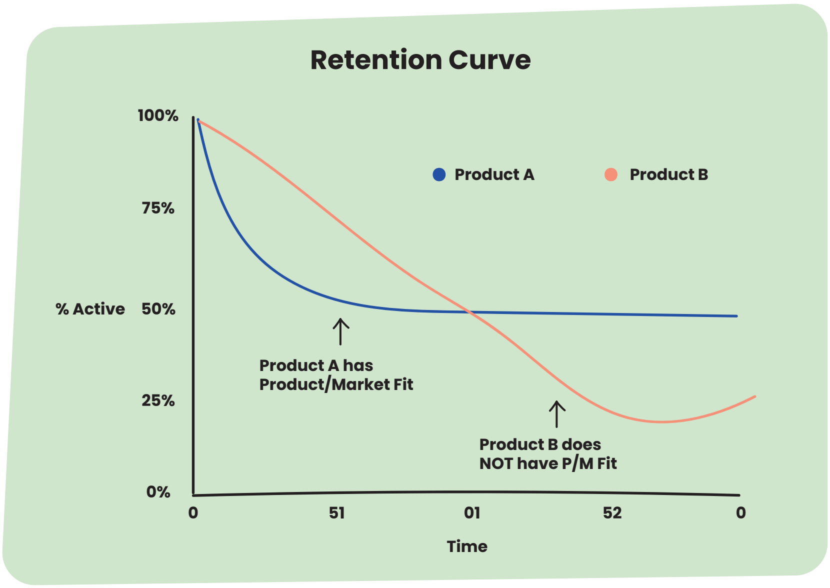
4. Review metrics monthly
You might be thinking, "why did you tell me to put my data front and centre if I'm only reviewing it monthly?" While you should live and breathe the data, you want enough time to pass in order to capture trends and themes. The variance when reviewing data on a daily or weekly basis is too high. Keeping an eye on your wins/losses is a great way to keep your team invested but allow enough time to analyse meaningful data. This will ensure you don't risk making knee-jerk reactions.

Being a product owner is a full-time job that only gets busier once the product is released. Keep this in mind when deciding on the team composition post-release. Ensure the product owner understands that feedback and data needs to be synthesised before being actioned.

Aiming for the 'Tick of Approval'

Whenever a product is first released it will begin with an active user rate of 100%. It's up to you how you define an active user but a popular definition is anyone that has used the product in the past week. Over time, that number must decrease. It would be unreasonable to expect every user to remain active.

The curve below is the retention curve you should be aiming for. While the active user rate falls dramatically at the beginning, you want to see signs of it slowing down. That means there is a core set of active users that use your product to solve a problem in their lives.



The most important takeaway from this diagram is that there will be user drop off, that is inevitable. Continue to concentrate on your core user base – those whose problem you're solving. They will be the percentage that remain active over time and create that 'tick.'

There is no universal threshold of active users which means product market fit. If you find less than 10% of users are active and there are signs the curve is still dropping then that is a tell-tale sign that either the wrong people are engaging with your product or it doesn't effectively solve the problem.

There are a few analytics tools that can help with mapping this data – see 3.2 Building in Metrics. Alternatively, you can find the user login data by querying your products database (provided your development company has set up the proper auditing functions). Look to your development company for guidance in this area.

Pivoting vs Iterating

You're probably familiar with the concept of pivoting. There have been so many success stories involving a pivot – one of the most notable is the story of Twitter.

Twitter was originally known as Odeo, which operated as a network where people could find and subscribe to podcasts. However, the founders feared the company's demise when iTunes began taking over the podcast niche. After giving employees two weeks to come up with new ideas, the company decided to make a drastic change and run with the idea of a status-updating micro-blogging platform.

If you've been reading this guide thoroughly you would have picked up on the concept of iterating. Don't worry, this isn't a test. Essentially, an iteration is a small change to a product, service or business which allows for value to be added within a quick timeframe. Each iteration is tested and checked before and after development, to ensure that the focus of the iteration meets the needs of the users and the market.

Sometimes, however, there are situations in which things have gone awry, and an iteration is not sufficient to course correct. In those situations, there are two ways of handling it: pressing on ahead into the howling dark, hoping that things will fix themselves, or a complete pivot. A pivot is a complete change to the product or service offering and may even involve a change to the underlying business model.

So then, when should you iterate and when should you pivot?

When to iterate

Iterations are the bread-and-butter of creating, launching and maintaining a product or service. Testing should be happening on a fairly regular basis in order to ensure that whatever is out to market is actually responding to the attitudes and needs of the users.

There are a few ways this testing can be carried out: analytics, user interviews, prototyping or ongoing market research, to name a few. Either way, a product owner can expect to iterate at least several times a year in order to keep their offering fresh. Iterations do not need to be major changes. Simple modifications, such as changing a font to make it more accessible, or tweaking an app's existing navigation, can make a huge difference to the user.

When user testing reveals opportunities to create value for users, this would likely indicate that an iteration of some sort would be beneficial. Ideally, especially for products which are already launched, iterations should be low-risk enough that they don't disrupt existing users, but provide enough value to justify the cost of development. In basic terms, iterative work is about staying nimble and ahead of the market by maintaining and improving a product in short bursts.

When to pivot

Iterations are fairly straightforward procedures, whereas pivots are much tougher undertakings. In general, pivots tend to happen when the core idea of a product or service has been undermined in some way. Taking a look at our Twitter example above, the very need for a place to find and subscribe to podcasts was threatened by iTunes. As a result, a pivot was necessary to create a new, valuable product.

When iterative approaches to better appeal or diversify the user base have failed, then the value proposition and problem statement may need to be re-visited. Ideally, the problem statement will never have to change. But sometimes things don’t go to plan. When the core of the problem statement changes, then you’re undertaking a pivot.

What this means
In many ways, a pivot means starting a project from scratch. This means that all of the work associated with initiating a project – market research, sketches, user flow diagrams, prototyping and the business case must be carried out in full. It’s much easier to iterate than it is to pivot.

A pivot should be seen as a last resort. If possible, try to pivot on the solution rather than the problem statement itself. Understandably this is not always possible (again, refer back to the Twitter example).
If you find yourself in a situation where a pivot is necessary then follow the theory outlined in this guide and try to pinpoint why the first attempt was unsuccessful.

workingmouse

Get in touch

Interested in learning more about software or think we might be a good fit for your next project?

(07) 3606 0230

growth@workingmouse.com.au

www.workingmouse.com.au