Designing and Running Experiment

What is MVP?

Minimum Viable Product or MVP is that version of a new product which allows a team to collect the maximum amount of validated learning about customers with the least effort.

MVP are primarily designed to test hypothesis and assumptions you have. That's why they are called "*MVP experiment*". It is a process and it relies on Validated Learning.

In validated learning you learn from customer in a test environment (does not bias customer, they act naturally, there is no pressure) whereas learning from customer is not designed as experiment hence you can't trust results because customers know they're being tested. Ultimately you want to know ***is our product going to be adopted by customers/user?***

Let's look at Zappos: when they started, they were not sure if people were willing to buy shoes online. So they build a very basic version of a website to see if people actually buy shoes online, they held no inventory. For their first orders bought shoes from shoe stores and shipped it to customers.

MVP is not prototype; MVP is about idea validation: building anything just to figure out if people are interested in what you are going to build. We build MVP to **mitigate risk** (time money and opportunity cost)

Heard about **fail fast**? This mantra comes from the speed you need to have in building MVP and running experiments in the same amount of time in order to collect more data – which means you will more likely find successful product as a result.

Why MVP is Important to Product Managers?

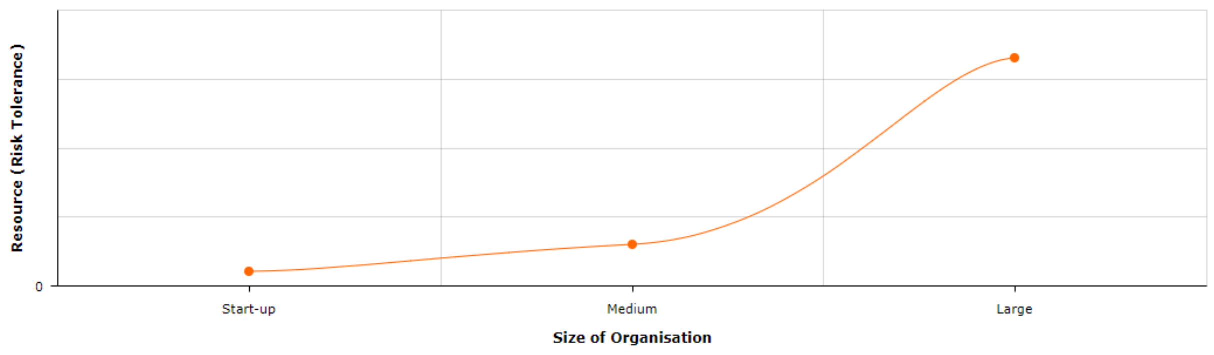
As everything in life, you have limited resources. As a product manager you have three main constraints: Time, money and opportunity cost. As a squad you really have 2: time and opportunity cost (can you work on something better in this time).

Let's see how this is different in start-ups and large organisations:

Start-ups don't have much money and have low tolerance for risk (if they launch wrong product, they will die like many do). They usually use all their resources to work on one product.

Bigger organisations care brand and opportunity cost. Start-ups don't have a brand (yet). Big organisations care about being first into a new platform or missing a big trend with one of their product lines.

Take a look at the example of Google Glass. They are less concerned about wasting resources or a fail a product (not that they necessarily want it but it is not the end of the world for them). For large organisations, the main risk is risk to their brand! For large organisations: **brand > resources**



This shows the bigger organisations get, the more tolerance they have for risk taking and the more resources they have, to try things out.

- *Think about where **your squad** sits, how much time and resources you have? Think about your portfolio and client.*

Seven Steps to Running an MVP Experiment

Here are 7 steps to come with MVP experiment, run it and learn from it

1. **Product solution ideation**

Figuring out what your problem and solution set is
What problem your solution is aiming to solve

2. **Identify assumptions with problem solution set**

Find the riskiest first
What are all the things that have to go right for this to be successful

3. **Build hypothesis around them**

Take assumptions and state what you think they are (build statements)
Establish criteria for testing them

4. **Establish MCF**

Minimum Criteria for Success
What constitutes success, what constitutes failure - define them for yourself
You need to have an idea if you succeeded or not

5. **Pick MVP strategy and type**

6. **Execute MVP**

Run your MVP experiment
Collect data
Iterate
Evaluate; what worked, what did not and what are you going to do differently if you want to run the experiment again

7. **YES/NO**

Hard yes or no whether it is a viable option or not

Identify Assumptions

In order to run an MVP experiment successfully, you have to identify all assumptions. Any new idea comes with a lot of assumptions, like anything in life we think we know things that are built on small things we assume is true, and those that are not true are going to become a problem! When you get into car, you assume the engine is going to start.

Write as many assumptions as possible (abstract process, there is no guide here). When you think people are going to like what you build; is it based on observation or intuition? Either case you assume!

Try to think about lean mantra and keep saying to yourself '**we don't know anything'**. **Anything you think you know, is assumption.**

In order for this problem solution to be successful, the following must be true:

- Enough smartphones for my particular target
- People trust this app
- Etc.

Here are common assumptions everyone makes:

- My user has x, y, z problem
- ________ matters to my customers (or users)
- Users will sign up (or pay) for this because we are solving X.
- No satisfactory substitute (many know how to solve it even without our or any tech solutions)

How to identify the riskiest assumptions

As a function of MVP and experiment, you want to mitigate risk, it makes sense to focus first on the risks that may sink the ship and make the entire thing unviable.

You don't want to spend a lot of time validating assumptions that are small and low-risk and after many iterations/experiments halfway through you realise you are going to fail because of the one big assumption you missed. You wasted so many resources because you started in the opposite direction!

Which assumption is the biggest deal if it turns out to be false? (that if turns out to be false, nothing else matters)

Let's take out example of most common assumption:

- My user has x, y, z problem

This is a big deal, if you build a solution for a problem that doesn't exist, you have to find the problem for your solution. And in most cases people can't do that which puts a big question on whether you should even do it or not.

- ________ matters to my customers (or users):

You may want to build something super-fast as you assume speed matters to your customers. But in reality, a huge improvement is needed in another area. Getting this wrong can impact how you use your resources but there is room to recover.

- Users will sign up (or pay) for this because we are solving X.

If they don't pay, there are other revenue streams like ads. May not be ideal but could work. But if they don't sign up at all or visit, that a problem

- No satisfactory substitute

There might be plenty of products and services that compete. People do not realise they have a problem unless someone says there is a better way! There is more than one way to solve a problem and sometimes your solution might be just over engineering (people use simple basic tools to solve their problems) Check Shark Tank video here on overengineering.

Having this in mind you have to design your MVP experiment targeting the highest risk assumptions first.

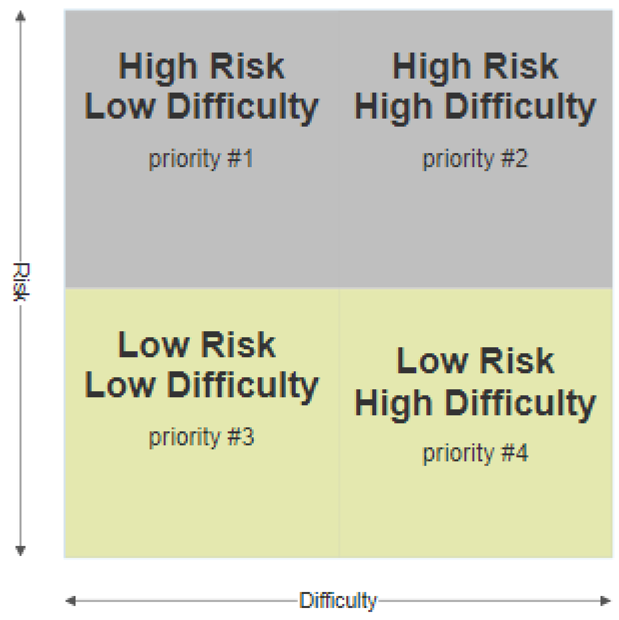
Risk/Difficulty Square

Product assumptions have 2 main criteria: risk and difficulty. As an entrepreneur you ideally jump on the riskiest assumptions first but as a product owner (or you working in a squad with limited resources and skills) you should consider both factors.

Large companies' mentality:
You are working on many many things, so the design team and dev team have to ration their time to decide what the task in hand is going to be.

Risks: how risky the assumption is to potential feature or product as a whole. Remember some can sink the ship and some change its course.

Difficulty: how hard it is to figure out this assumption is real (how much effort you have to channel to testing this)



Be on lookout for low difficulty (assumptions that are not resource intensive) with great ROI (addressing highest risk assumptions).

What is hypothesis?

With your list of assumptions, your goal now is to steadily and methodically test assumptions and roll them out.

By now we have developed an assumption list i.e. raw list of things we roughly think need to be true for success of our product. They are not precise nor **actionable**. They must be testable and concrete not abstract. E.g. 'People are satisfied parking in a garage' is not testable!

What is a hypothesis? **It is a single written testable (actionable) statement of what you believe to be true with regards to assumptions.**

For example, the assumption that: people are not satisfied parking in a garage is not testable! We need to construct a test around the statement above. We need to be more specific. E.g. Who is not satisfied? How unsatisfied? Why?

Developing a product or feature is entirely a scientific process and a science experiment. This brings clarity to everyone in the team; what needs to be done, in order to achieve exactly what (should be measurable), how much effort is needed, etc.

Putting together a hypothesis

Remember the main difference between assumption and hypothesis is that **hypothesis is actionable**. There are many ways to translate an assumption to hypothesis. Here are few techniques to do this.

We believe __________ will __________ because ____________.
We believe *the target/group of people* will *predicted action* because *reason*.

You can add problem/benefit to your hypothesis. This format is very basic because you are just trying to establish if people are interested in your product. This is an exploratory test as an entrepreneur but if you are managing established product or in your squad your client says that is the case) you have to be more specific.

This could be small tweaks and changes to existing products.
If we ___________ we believe ___________ will ___________ because ___________.
If we *action* we believe *subject* will *predicted action/outcome* because *reason*.
e.g. if we change the colour of Buy button, we believe users will buy more (MCS) because they are busy and get distracted.

So let's look at the better version:
We believe __________ has a __________ because __________ . If we __________ , this __________ metric will improve.

We believe *subject* has a *problem* because *reason*. If we *action*, this *metric* metric will improve.

What is Minimum Criterion for Success?

Remember the outcome of MVP test is 3 main things:

1. Your hypothesis is categorically false and not worth doing
2. Your hypothesis is true with a wide margin so you will be fine building the product
3. You are somewhere in the middle (99% of MVPs end up here)

That is why MCS becomes important. It is defining that MORE in your hypothesis. It may increase sales or speed by x percent but is it worth doing considering organisation size, resources like number of people in your squad and time you have (12 weeks?) Maybe it is if it is an increase by 60%!

If you forget to set your MCS, you might end up conflating a validated hypothesis with a clear signal to proceed. They are different. You can validate a hypothesis in a way that you do not build everything! MCS gives your experience clarity and meaning, we are not trying to just improve stuff. We are trying to improve x by x amount or percentage.

It is fairly simple to do MCS, **starting from right**, try to think about metrics that needs improvement, then all cost associated to them

Cost (of making change or building new things)	Reward (metrics) should signal success and be viable
Development time (money or loss of time) Your time Legacy issues Opportunity cost Brand effect	Increase revenue Time spent on page Conversion rate Sign up rate

Now **starting from left**, try to see how much we are spending for this to happen.

Example:
Development time (could this time be used to fix existing issues)
Run ad to announce the feature (not relevant in your squad)
Opportunity cost (needs careful consideration in your squad, can you use this time to do something else?) what else can you build in this time?

Now on right side:
Focus on one metric
Thing about past metrics and what they mean
What needs to be moved by how much

It is very hard to get this number precisely even for established companies and product managers. The whole point is you want to coordinate your activities and align it to a bigger goal in organisation. You want to know how much a metric needs to be improved for your team to make sense to spend that time and resources doing it.

MCS for start-ups

As a start-up you are going to be more concerned with certain metrics and less sensitive to certain risks and costs. For example, average minutes spent is not important if you do not have any customers!

In start-up you need to focus more on **validation metrics**. Things like:

- Is my product a good idea?
- Does it meet user needs?
- Does the identified problem really exist?

Validation metrics are metrics that demonstrate real interest from potential customers. They are all measuring interest not behavioural changes. Things like:

- Percentage of people sign up
- Percentage of people share your post
- Average purchase price

- *Think about where **your squad** is, what are the valuable and applicate metrics for you? Have a conversation about this with your client*

MVP techniques:

At this stage we know what we want to build (ideally), we also know what types of assumptions we need to test to get to the truth and what outcome we want to see out of these MVPs. When you are running MVP experiments, you need to imply what we are presenting is already real or coming very soon (for some of the MVPs). We need to do this because we need "validated learning". We need users to be in real life situations where they don't know they're being tested and behave normally. But we might not have decided (internally) what we are going to build. So sometimes we have to fake things to a certain degree. These techniques below are in order of how much you have to fake your MVP Experiment.

- These techniques are industry-based techniques and MVPs for validation idea. **Your squad** works on a prototype or a functional MVP. It is good to know about other techniques. You might be able to use some of them partially in your product (to be consulted with client)

Email MVP:
All you need for email MVP is list (to email to) and some wordsmithing skills. Take a small segment of your list and email them a pitch for the new feature for the product and see how they react. This is not the best idea for going about MVP but is an available option some product owners use. This is really for the pre-product stage. AppSumo is one example, they sell products via email before building it.

Suitable for:	small organisations and start-ups that do not have brand anxiety
Pros:	Quick and cheap Limit testing group Segment users (if you have secondary data on them) No development needed
Cons:	It can come off sloppy It could dent your brand

Few tips:

- Be aware of what the audience expects of you. Different audiences have different tolerance
- Match the production value of your company's typical email (tone, images, design). Use Canva
- Try to pair this with a landing page or another MVP technique like landing page or concierge service

Shadow button MVP
This requires more work compared to email MVP but still very easy to do. In an existing product, you put a button that supposedly links to a new feature. When a person clicks on it, it registers that someone clicked on it and either does nothing or shows text saying it is 'coming soon'. People act differently when they think something is real versus when they have a reason to think it is not real. This is mostly used in mid-size start-ups when they try to triage their 'nice-to-have' feature. E.g. when you try to figure out which log-in people prefer to use (Facebook login, LinkedIn login, Google etc)

Suitable for:	Medium size start-ups When you want to decide what option is more popular among users
Pros:	They give you great data Very easy to do
Cons:	It does not look good (looks broken) Sneaky

Few tips:

- Acknowledge and thank – intentional bug is better than bug you are not aware of! Think about negative image it creates in the long terms.
- Limit the amount of users that can see this if possible. Calculate the number of people who need to see this to be statistically significant

404 Page/Coming Soon MVP
You act like you are adding a new feature or product and when the user navigates to a new page it either displays 404 or coming soon (followed up by sign up, or pre-order or register your interest). This might not seem professional to you but guess who uses this a

lot? Amazon! They use it extensively to find out if users are interested in their side projects or find better ways to organise their categories.

Ask yourself as a user what is less evil? A broken page or misleading page

Suitable for:	404: Organisation with lots of pages as users can easily go to another page without being affected much. Coming soon: companies whom audience know they are new and consider themselves as early adopters
Pros:	Easy to set-up Easy to get analytics
Cons:	You have a broken or misleading page. You must manage the effect of this on your brand specially if you are a start-up

Few tips:

- Design it, do not use generic template and make sure you proofread
- Launchrock and KickoffLab are two good tools for designing these
- Shorten the horizon. Do not do it too much

Explainer MVP
A video that explains what an app or feature does. It is either tutorial style or sales style.

In tutorial style someone explains how to do certain tasks in their product and why they made it (they haven't actually made it). It is smoke and mirror!

In sales explainer you essentially pitch a product or feature in a promo (they aren't actually built yet). You just add something like 'sign up for notification'. And based on how many sign ups you get you decide whether you should build it or not.

Suitable for:	Companies who frequently use these videos Companies with complicated products Audience that are used to this and the infrastructure and resources are already there
Pros:	Video converts much better than just text Provides in-depth explanation They can excite users really well
Cons:	Needs resources; videographer, editors and custom graphics

Few tips:

- Consider your size and user expectation: if you don't have strong presence you can be scrappy with the video but as an established brand, your audience would have a different expectation
- Tools: Wistia (has analytics), Powtoon, GoAnimate (now Vyond)

Fake Landing Page/Pitch Experiment MVP
This is when you create one singular page of all features of a product and add a Call to Action in the bottom of the page

Suitable for:	Start-ups High level idea validation
Pros:	Low cost Can direct traffic by Google Ads or other means
Cons:	Lots of thoughts needs to go into the content and graphics

Concierge MVP
In this MVP you launch informal offerings into a small set of users. In their eyes this is the Beta version. You help them manually accomplish a task they are trying to do so you can see firsthand what you are doing is helpful and necessary or not.

For example, if you want to build a bot, start with some power users, tell them you are launching a new service and ask them what they need help with. Come back with personalised recommendations and help them via email. See if they engage and ask for more. Try to understand what exactly they use the service for.

Suitable for:	When you manually want to help users accomplish their goals as a means of validating whether or not they have a need for what you are doing. You are transparent that you haven't built it yet.
Pros:	No buildout required, you don't even have to add fake buttons Can be run almost in secret (you email people and manage relationships directly) It's a hybrid approach between customer development and MVP testing
Cons:	Management intensive and time consuming Logistical issues (as you are responsible) A lot of resources on per customer Very expensive to run (your time, customer time, follow-up)

Piecemeal MVP
Instead of building a product, you take what's out there in form of out of box available software and by piecing those software together you can match the functionality you need to test the basics of what you want to build.

Suitable for:	When you want to add functionality to your product that is common
Pros:	Reduces the development time Could possibly be cheaper
Cons:	When you have multiple functionalities that need to integrate with each other

Few tips:

- As a product manager, you cannot get away from sending people from one software to another. You need to find white labelled products or those that can integrate and imbed into your product
- Tools: Zapier, IFTTT, Formstack, Typeform, JotForm, Twilio and Recurly

The Wizard of OZ MVP
From the front-end this looks to be completely made but all the things supposed to be working and carried out by code and computer are actually carried out by individuals behind the scene. The biggest resources you are going to spend is building the business logic that no one is going to see. So you take advantage of that in front-end and build only front-end

Suitable for:	Companies with ability to assign people do manual work Business logics that are simple enough to be performed by human
Pros:	Less work for development team In users' eyes it is fully built product
Cons:	Resources must be allocated to manually do the work Can only work if the logic and business process is simple and manageable by people

Evaluating result and learning from them

We all know the MVP process now

- Come up with idea
- Design experiment
- Set up expectations
- Run a test
- Gather data and compare with MCS
- Make the call
    - Make the feature and give it a green light
    - Scrap the whole idea
    - Re-run the test on different hypothesis

Interviews are going to give us qualitative data (feelings and verbal). MVPs are going to return quantitative data - it has numbers: percentage of users, how much time on x, total sign up, etc. You need to collect both types of data to make right decision: Numbers to know what happened (unbiased) and interviews to know why.