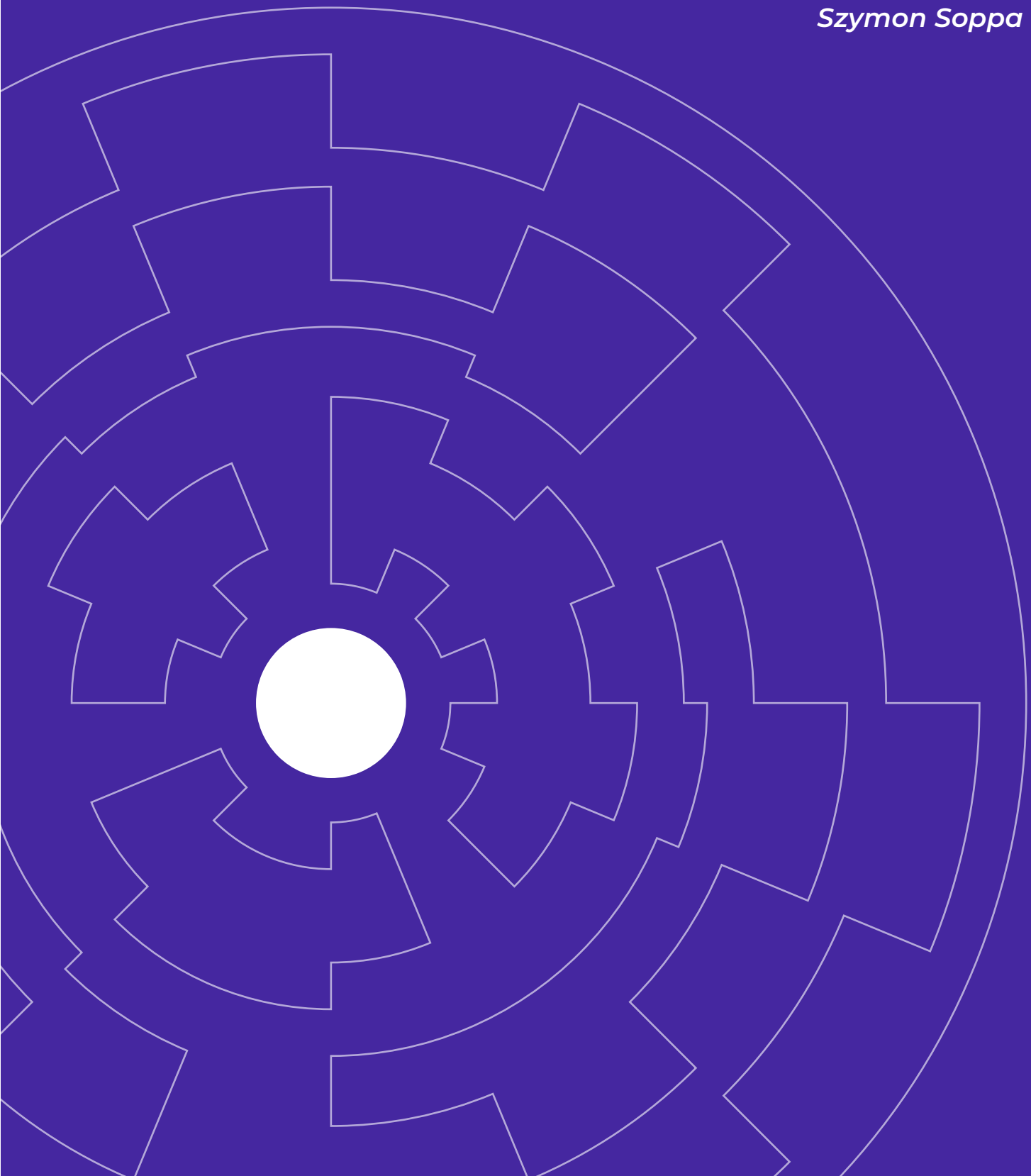
CURIOSUM **E-BOOK**

MVP Builder's Guide for Startups

Szymon Soppa



Index

Nobody is born
an entrepreneur.

**We are all made in fire
of trial and error.**

Author

Szymon Soppa

Experienced developer, founder of ***Curiosum***. He has led a diverse package of development projects of all sizes, especially in entertainment, recruitment, e-commerce and e-learning industries. Szymon created a start-up of his own in the entertainment industry.

curiosum

Curiosum is a Software Development company, specializing in development outsourcing projects and bringing partners' ideas to existence.

drop us a line!

Introduction

Szymon Soppa

“

Most of our ideas are less than great. But some deserve special attention, work and maybe even realization.

If you already have that type of an idea in your head, we are here to help you with the next step. Let’s build it!

But to succeed with that, you need the right mindset and the right tools. This is what you will find in this e-book.

What will you will learn from this ebook:

Why MVPs matter?

How to prepare for creating an MVP step by step?

Why creating your MVP is just the beginning?

"Visionaries are especially afraid of a false negative: that customers will reject a flawed MVP that is too small or too limited.
[...]
The solution to this dilemma is a commitment to iteration. You have to commit to a locked-in agreement—ahead of time—that no matter what comes of testing the MVP, you will not give up hope."

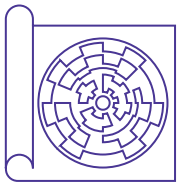
— Eric Ries, ***The Lean Startup***

What is an MVP?

Are you sure you actually want to make an MVP? Take a look at the key differences between a Proof of Concept, a Prototype and an MVP:

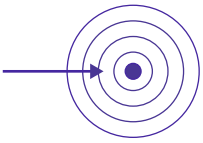
Proof of Concept

Technical feasibility of your idea.



Prototype

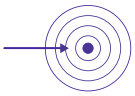
How will your idea be built? How will it look like?



MVP

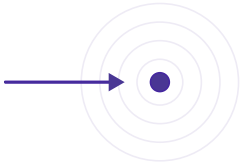
Limited, but functional first deployment for user validation. Do users like it?

Misunderstanding the stage your idea needs to be at leads to a longer and more costly development cycle than necessary. If you aren't sure if a difficult feature will be possible to code and include, maybe go back to the PoC. If you have not thought out what the look and feel of the MVP may be, finish a prototype. You will find a more detailed breakdown below.

	 *POC*	 *Prototype*	 *MVP*
Tech oriented	✓	✗	✓
User oriented	✗	✓	✓
Support growth	✗	?	✓
Enables finding investors	?	?	✓
Budget			
Business value	Is core feature possible?	Am I generating user value?	Real market feedback

How to think about your MVP?

MVP is NOT supposed to be just a crappy first version!



It is all about idea validation.

It should start your learning process as fast as possible. MVPs do not have to be very small – but just be big enough to study feasibility of your idea using minimal effort.

Traditional product development usually requires long, thoughtful incubation period and strives for product perfection, but the goal of the MVP is to start the process of learning, not end it.

MVP is the start of a classic Build-Measure-Learn cycle. Such approach mitigates the risk in you upfront investment and enables you to run multiple experiments at once.

This is why start-ups use the MVP concept a lot – they must keep the risk low, and the resources to burn are not plentiful. Larger organizations however can simply fire off multiple MVP-calibre tests at once to cover more market.

Need some assistance?
Schedule a free 30 min consultation for your app idea:

contact us

Before the code is built

You better have a plan! Here are your basic steps to a shiny, functional MVP:

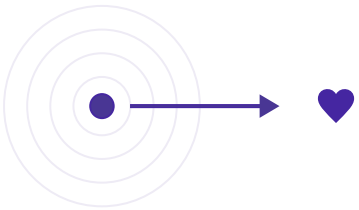
1. **Understand why you are doing an MVP**
2. **Set up specific and measurable goals**
3. **Research the users**
4. **Create a User's Journey**
5. **Agree on the list of features**
6. **Choose the tech and/or your development partner**
7. **Execute!**

A full, detailed MVP Building Checklist can be found at the bottom of this document.

Not launching = painful
Not learning = fatal

6 reasons why you need MVP

The biggest risk is that you create something that nobody wants. Therefore, your goal is to find and test specific assumptions about your product. Dozens of top apps, like Uber and Airbnb had only bare-bones versions of their apps to start off with. But crucially, the value in their MVP versions was enough to attract.



There are 6 core areas where an MVP helps:

1. **Generate stakeholder/investor buy-in**

Make them see your idea at work! Finding a willing investor is much easier when you can show, not just tell. Same with internal stakeholders – find more traction with your managers and executives with an MVP that works.

2. **Test business concepts**

As a useable prototype, MVP allows you to release such a version to the prospective users and get data on their interactions and sympathies. Remember to have a strategy of gathering such information in a structured way! Remember that you can only test SOME business concept at one time. This is why, as Eric Reis said, you should remove any feature that does not contribute to learning what you want to learn!

3. **Verify market demand**

MVP gives you more options for measuring market demand. Did the users enjoy this feature? Who was willing to pay for the MVP, and who wasn't? Which price plans fared the best?

4. **Develop monetization strategy**

Thanks to your new data, lessons should be learned. Which path should you now choose? What is my ultimate commercial goal? What can be realistically achieved?

5. **Test UX and usability**

You finally gave the real thing to the users. Have all your design people been wrong? Or were the interactions pretty frictionless and clear? Ask a good number of users if you can, it's the best time to do this!

6. **Cost efficiency**

Keep your time and costs down and avoid extra features. Learn what you can spending a little, and be ready to leave the idea if your well thought-out MVP test run flops.

To sum up, you should prove or disprove your assumptions, keep costs under control, development time short, and also keep reworks minimal.

All this gives you power with the stakeholders.

Setting the right goal to succeed

Now that we know exactly why we create it, let's start designing our MVP and build it! In this stage, your goals should be related to four key aspects of building an MVP:

- **Finding the right problem (niche) to solve**
- **Finding the right group of target users**
- **Setting up long-term and short-term goals**
- **Setting up success criteria (KPIs)**

Goals for your MVP usually should not be simply scaled down version of your final product's KPIs. Those are categorically different and you should shape expectations accordingly. One idea is to evolve the KPIs along the developing product. A great example is described *here*.

Who are the users?

This is the moment to create good User Personas. There are dozens of ways to do this. You will find good guides with templates *here* and *here*. User Personas will be useful in the next step – when we will be creating a full User Journey to visualize the entire map of possibilities.

Why create a User Journey?

It allows you to:

- **Define and visualize priorities better**
- **Create alignment of your objectives**
- **Put users squarely in the centre**
- **Have better and more useful discussions about the product**
- **See if your User Personas make sense**

There are multiple ways to create the User Journey. It is a visual overview of the entirety of your customers interactions with your future product. It does not only contain specific steps a user takes, but also the accompanying emotions and experiences the interactions provoke.

You should define these aspects of the journey:

1. **Stage**

   General state of mind of the user. Is the person aware of our services, or still has to get to know them? Is it the post-sales stage of building loyalty?

   For mobile applications, for example, the stages could be: Awareness, Consideration, Installation, Usage, Deletion.

   For a web service: Awareness, Consideration, Purchase, Onboarding, Advocacy.

2. **Specific user actions**

   Those can be for example: Seeing an Ad for your product, or watching a YouTube video about it, later specific in-app actions, like using a feature.

   Sometimes it will be necessary to separate all relevant touchpoints where users interact with the product.

3. **User goals**

   What goals were behind this action? What was the user trying to actually accomplish?

4. **Storyboard**

If you want, draw (or task someone who can) simple representations of the action and the user for purpose of easier visualization and conceptualization

5. **Experience**

What emotion or thought occurs when the user does X or Y. Does using this feature change the mind state of the user? Why?

6. **Problems**

What are the issues a user may have at this stage? What frustrates her/him?

7. **Ideas**

What can be done at this stage to assist the user and alleviate her/his pains?

Remember

There are multiple competing frameworks that dictate how to segment the User Journey. If you do not have experience in this, keep it basic! Three stages – Awareness, Consideration and Decision should be enough for majority of MVP user maps.

The journey should be more or less linear – it must have several starting points, a group of actions in-between and a story ending.

Choosing final MVP features

Now that you understand the user and their journey better, it is time to finally set out the final set of features your MVP is going to have.

Choose features

The features should solve the most crucial pains and problems you found writing down the User Journey. Brainstorm a lot here – there are going to be many combinations that seem to make sense, but you are probably going to make a single MVP prototype. Prepare a broad list, and then narrow it down to a necessary minimum.

Define and break them down

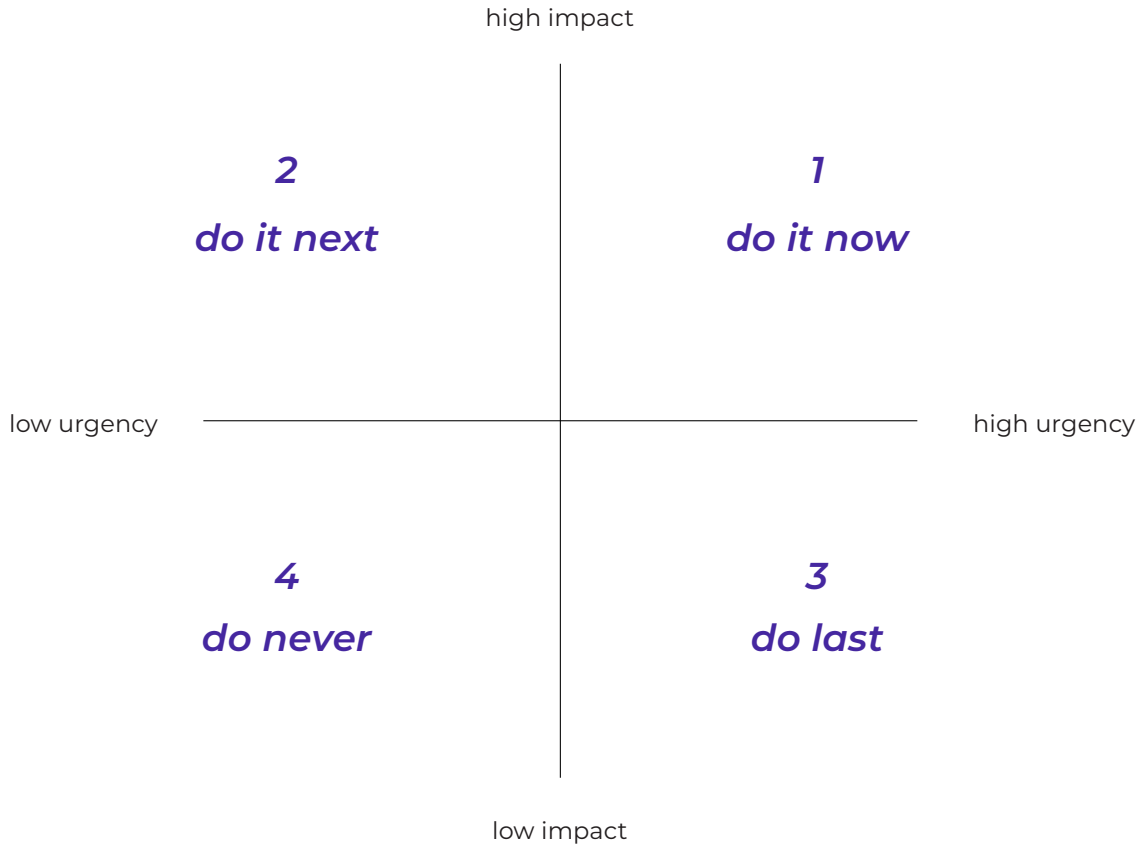
Next, you need a detailed, high resolution description of the chosen feature and what it has to accomplish. How will this feature be constructed, and what data is necessary for this feature to run?

Prioritize features

Your features are all awesome! But not all of them are necessary now. There are specific ways you can go about prioritizing features at this point:

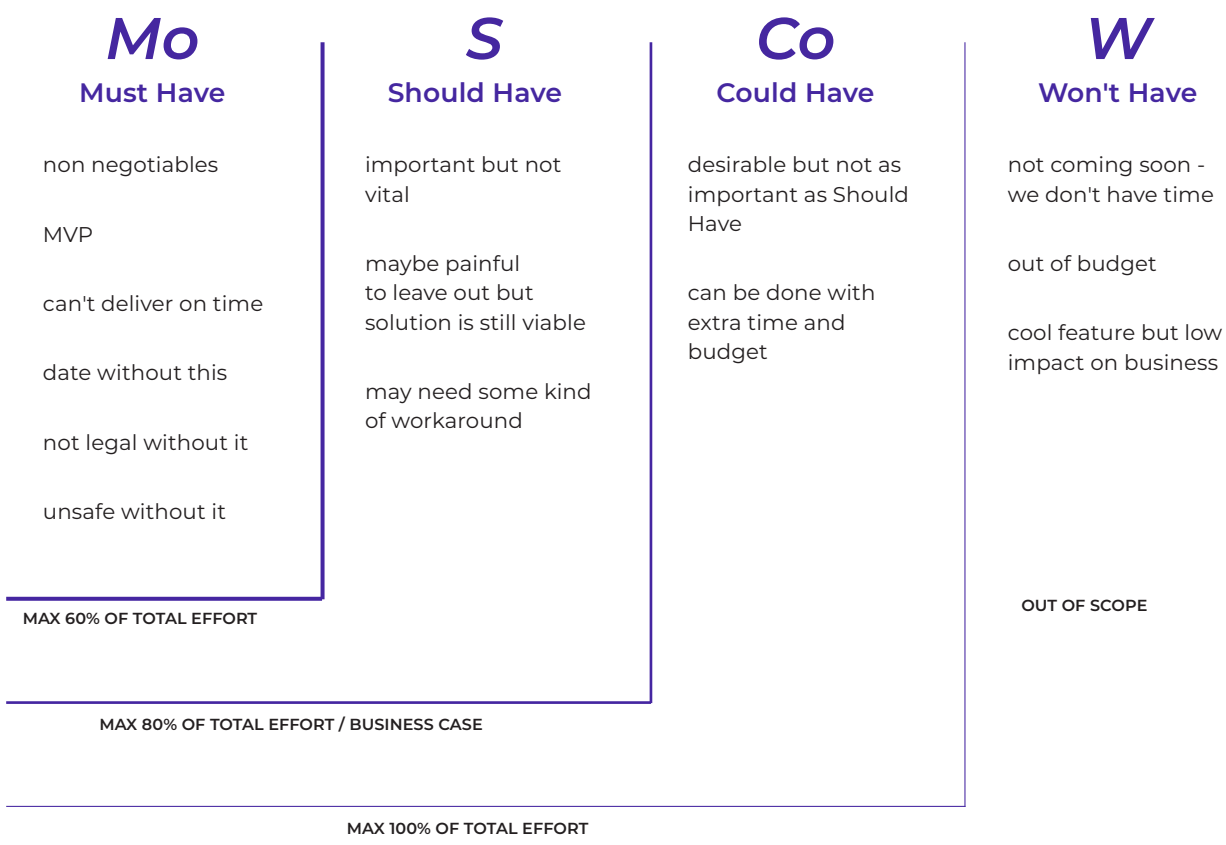
1. Prioritization matrix

There are a couple types of such matrixes. A simple version, with urgency and impact on the X and Y axes', is shown below:



2. MoSCoW method

Do you think you must or just could have this feature? Break it down using the MoSCoW method, with the example depicted here:



Must Have

Ask yourself: "what happens if this requirements is not met?". If you would have to cancel the project, this feature is must-have.

Should Have

It's about degree of pain, business value lost and proportion of users affected. If those are really low, it's not a Should Have but a Could Have.

These tools enable you to settle on an eventual MVP scope.

Choosing the technology

There are key considerations to watch for here:

1. If the product is a part of broader technology ecosystem, its interdependencies may (and probably should) influence your decision on what technology should be the backbone of your MVP.

2. Pick a technology that achieves your goals fast and cheap, or pick developers you trust and let them guide you with the technical aspects of the MVP creation.

   There are three key issues that are often mutually exclusive – there are no perfect coding languages.

   - **Scalability** (our picks for scalability: Elixir, Go, Scala)

- **Costs** (our picks to bring down costs: Elixir, Python, Ruby, JS)

- **Versatility** (our picks of languages with rich library of features: Java, JS, Python)

Ultimately, sometimes choosing the tech stack is less important than first-time start-up founders imagine it to be. The level of expertise of the developers, however, can make or break a project. So, look for companies and team members that have worked in this language a lot.

Find out which top companies are using Elixir – and why:

visit our blog

Promote before you launch to avoid a flop!

Let your prospective users know something is coming! Using services like *ProductHunt*, *BetaList* and even listing your upcoming product on topical subreddits should help generate early adopters.

Differentiate between Soft Launch, Dark Launch and Hard Launch!

Soft Launch: to reduce risks and exposure, release your product to a limited number of users. For example, release it for a single geographical location or a niche audience. This allows to test the monetization options and receive customer feedback at a lower cost.

Dark Launch: this is a process of releasing the product to a select group before the official launch itself. Essentially this is expanded post-beta-testing.

Hard Launch: this is going all in! If you feel that you know your audience very well and are ready to take that risk, hard launching is immediately starting strong promotional activities on a large scale. This option usually does not work for smaller enterprises.

Build, Measure, Learn

(Or maybe the other way around?)

The Build-Measure-Learn cycle is at the heart of modern lean, agile start-up development. Some cycles will last a couple of days – other however, months or years. The BML loop is frequently misunderstood, so let's dig into it a bit deeper.

First, you do not have to use this loop as your frame of reference. There is *Think-Make-Check* in Lean UX and *Plan-Do-Check-Act* in Quality Control. Find a loop that is understandable to the organization and that reflects your approach and values.

Reverse it!

As we shown you already, there is a fair bit of planning before you actually build for the first time. Remember that “Building” serves an important purpose, which is to test hypotheses about problems your users may have and want to solve with your product.

This is why we believe in the reverse version of Build-Measure-Learn!

There are three steps to this:

1. **Identify the Learning Goal**

   We got this covered above – but in general, you need to learn what you need to learn first. Prepare some hypotheses, and the design an experiment to get the necessary data.

2. **Specify the final Data**

   What exactly should the numbers look like after you are done? What metric will you be taking into account, and which ones are irrelevant at this stage?

3. **Build it**

   Only now you are ready to throw some real time and effort at the problem.

If you are outsourcing the development, enjoy your free time – but always be a proactive, communicative partner to the software house you are partnering with. Ask them multiple questions and allow the developers to have a voice in the process.

If you are working on the product 100% internally, ensure that the development team has enough time and managerial buy-in to create the product without too much disruption.

After completing a single loop, you should be armed with new insights – both about your product, and the way it should be developed further.

MVP Building Checklist

1. Management & Governance

There is buy-in from relevant stakeholders.

You understand who your stakeholders are and what do they need. You know what information you need to give them and at what time.

A structured decision-making process is present. Roles and responsibilities are clear to all engaged.

There is a plan what to do when the MVP actually succeeds.

2. Financial & Legal

- [ ] You have a good name for the product.
- [ ] You understand the legal risks of that particular type of product.
- [ ] You understand what needs to happen for the product to be financially sustainable (a business model).
- [ ] You have a draft pricing structure
- [ ] You have a detailed budget for a relevant period (all costs and revenues)

3. Growth & Marketing

- [ ] You know your marketing budget
- [ ] You have basic visuals done (logo, colours, fonts)
- [ ] You have your core message and value proposition set
- [ ] You are ready to promote your MVP long before launch
- [ ] You have a detailed marketing plan

4. The product

- [ ] You have researched the competition
- [ ] You have created your User's Personas and you know their problems/pain points
- [ ] You have created the User's Journeys
- [ ] You have decided on the list of features MVP will include
- [ ] You have chosen a tech stack
- [ ] You have chosen an outsourcing partner or set up a team in-house
- [ ] You understand how the product will fit into the existing company structure and its business
- [ ] You have a detailed project timeline
- [ ] Key stakeholders are in agreement about the points listed above

5. Goals & Data

- [ ] You know what you intend to track
- [ ] You know which data and metrics needs to be collected
- [ ] You know how you will analyse this data
- [ ] You know what are the success criteria
- [ ] You know what are the failure criteria

6. Launch & Vaildation

- [ ] You have a launch plan
- [ ] You have your feedback gathering systems in place
- [ ] You provide adequate user support