ATLAS
_CLEVER SOFTWARE

WHAT ARE MINIMUM VIABLE PRODUCTS?

SIMON SWORDS
Managing Director, Atlas Computer Systems Limited
simon.swords@atlascode.com

Last updated: July 2018

CONTENTS

INTRODUCTION

The Internet is littered with failed companies that splurged insane amounts of money on their new software start-up, only to discover too late that their product was never going to succeed. Equally, large companies attempting to get ahead of their competitors or spin out new start-ups are equally exposed to the risk of their software projects going sideways – perhaps even more so.

Simon has launched a range of SaaS businesses in both the SME and enterprise spaces

The list of reasons that software projects fail is seemingly infinite but the scenarios I commonly see include:

1. A lack of an addressable market
2. A flawed business model
3. A lack of a product/market fit
4. A software solution that solves a problem which didn't exist in the first place
5. The software gains new customers and traction immediately after launch but customer churn outstrips customer growth because the software isn't "sticky" enough.

I've launched dozens of software solutions over the last decade, and as the **owner of a software company** I understand how exciting the whole process of software development is. With this excitement, comes the opportunity to get carried away on the design and build of a software masterpiece. You want all the bells and whistles to be added from day one because unless your software solution is the Swiss army knife of software solutions, how could it possibly succeed?

I know this feeling, because when I first started building software solutions I made the same mistake.

Avoid failure by limiting your exposure to risk

At some stage we've all thought we had the idea for the next **Facebook or, we plan to show Jeff Bezos a thing or two about how to run an ecommerce website**.

Sadly, all things being equal, your software business idea is more likely to fail than it is to succeed. **Forbes** puts the number of start-ups that fail at 90%. So it is essential that you perform the technical equivalent of dipping your toe in the water before you throw everything you have at your new software project.

A Minimum Viable Product (MVP) gives you the chance to do just that, and in this guide I'll explain why and how you should take this approach with your next software project.

MVP

What is a Minimum Viable Product?

An MVP is a software product with a basic set of features that are just sufficient enough to capture the attention of early adopters and solve one or two very specific problems for those people incredibly well.

I've got funding – the MVP approach doesn't apply to me does it?

Congratulations! You've got funding and what feels like a bottomless pit of money to throw at software development. You now need to consider using the MVP driven approach to your software project even more than those who are self-funding their project.

Finance backed companies need to be even more ruthless about how their initial software development is prioritised. With deep pockets and all the resources that come with them, the constraints that might naturally keep the initial software product streamlined and focussed aren't there.

There are of course exceptions, but my feeling is that most software projects need to take the MVP approach for their initial product launch.

Why use a Minimum Viable Product?

Most businesses start with a person who has discovered a problem and has created a hypothesis for what a potential solution might be.

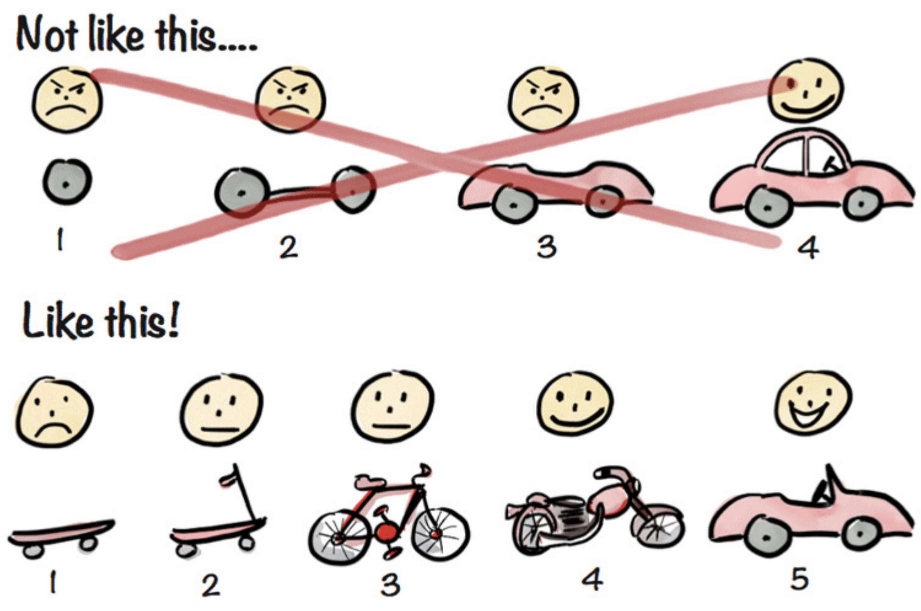
Steve Blank defines start-ups as "an organisation formed to search for a repeatable and scalable business model".

When you think of it like that; the less time and money you spend on each search the more searches you can carry out. If you maximise the number of searches for the right customer solution in your new start-up, you maximise the chances of finding success.

The MVP approach to building and customer testing a software solution is a smart way to:

1. Solve one or two problems for a specific target audience
2. Release your product to market in the shortest time
3. Invest the smallest possible amount of upfront time and financial investment
4. Test the demand for your product – before releasing a fully-fledged product
5. Gain valuable insight on what works and what doesn't work from actual customers

MVPs help you learn more about your product and gain customer feedback. You release early, learn immediately from people who are qualified to provide feedback (your target audience), and iterate quickly.



Source: Henrik Kniberg

NOT LIKE THIS, LIKE THIS!

I like to use the above graphic to drive home exactly how a MVP driven project works. In this graphic, the customer has requested that you build them a form of transport that will get them from A to B.

As a quick aside, I'm using the term "customer" here as a generic term to represent anybody who will use the first version of your product.

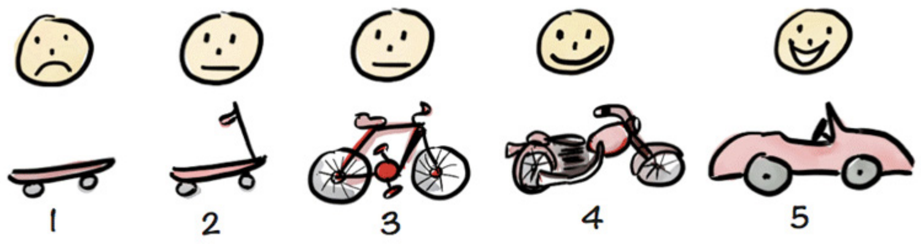
Not like this

The top row of the graphic represents the non-MVP approach. This approach is focussed on the end goal of building a solution for the customer that gets them from A to B. Without the MVP mentality, the engineers behind the solution conclude that there's no way the customer would be happy with an unfinished product – so they continue to build until they're 100% complete and a car is ready.

With each new delivery, or iteration, of the proposed solution the customer is left more annoyed and angry because despite being shown the work in progress, they can't use it. A part completed car can't be driven, is therefore untestable, and the customer cannot give any meaningful feedback as the solution takes shape.

Even though the customer is smiling in step 4, there's a high chance she's going to find something that she isn't happy with in the final build as she hasn't been given the opportunity to test and provide feedback on the car as it's taken shape.

All in all, this process is designed to fail.



Like this

Here we take a very different approach.

We start with the same problem – the customer wants to get from A to B.

However, instead of focussing on building a car which is arguably the optimal solution – we focus on the underlying need to get the customer's problem solved ASAP and obtain their feedback. A car is one possible solution to that, but perhaps in the first instance there is a solution we can provide the customer that solves the underlying problem without all of the cost, time and risk associated with building a car on day one?

So we build the smallest, easiest and least risky solution possible – and in this example that's a skateboard.

The customer won't be completely happy with your solution, especially if they're travelling a long distance. However, in the absence of another solution they will use the skateboard – and more importantly they will give you feedback and you will have the **opportunity to learn**.

Unlike the top row in the graphic, the skateboard is a basic, minimalist solution to the problem presented by the customer. The message to the customer in this scenario is "Hey, we're working our way towards the car – in the meantime can you give this a try and provide us with some feedback?". The goal is to think big, but deliver incrementally.

You can see in the second step that we've taken on-board the feedback from our customers who tried skateboard for a while. They liked the skateboard but sometimes they would fall off, so we added handlebars to help them to remain stable.

Step 3 is the result of feedback from the customer that for long distances the skateboard with a handlebar (aka scooter) just isn't fit for purpose. Okay no problem, we take that feedback and after a few meaningful customer conversations the outcome is that we build a bicycle.

The customer then complains that sometimes they need to travel cross country and the bicycle just isn't fast enough – plus it makes their legs sore on long journeys. No problem, step 4 sees us create them a motorbike so they can travel longer distances with ease.

The motorbike is a huge hit, but if only the bike could carry 4 people and had a cover so that when it rains they don't get wet. Which brings us to our final stage, step 5 – the delivery of a car.

The customer is delighted. They've had a usable product since the first build (the skateboard) and they've had a say in the development of that skateboard to bring about the final solution – their car. Not only has the customer had a solution to their problem the first build – their input has shaped the final car to include benefits and features that fulfil their exact needs.

In summary, what's your skateboard?

The top row scenario of delivering a front tyre is sub optimal because we keep delivering features that the customer can't use. They're frustrated, they can't provide meaningful feedback, and the end product might meet their general needs but won't be as refined.

So take the MVP approach instead. Specifically, think of a skateboard as a metaphor for the smallest thing you can put in the hands of real customers, and get honest and usable feedback as quickly as possible.

This will give you the vitally needed feedback loop, and will give both you and the customer control over the project – you can learn and make changes, instead of just following the plan and hoping for the best.

HOW?

How do you Build a Winning Minimum Viable Product?

If you have an idea of a product but you aren't sure if you can successfully develop and launch it, an MVP will help you reduce risk and find out early on if your idea will work.

When it comes to building a minimum viable product, keep in mind that it should be minimal. Keep things as simple as possible from the very beginning.

Here's the minimum viable product template process when you engage with us via our Atlas Roadmaps Workshop.

1 IDENTIFY YOUR TARGET AUDIENCE AND THE SPECIFIC PROBLEM YOU'RE SOLVING FOR THEM

Take a look at your business idea and ask yourself why you need this product and identify how it can help you. This will help you identify the main goal of your product and the solution to your potential customers' needs.

Speaking from my personal experience, I tend to build the best products when I'm trying to solve a problem for myself, aka scratch my own itch.

You need to think very hard about why customers would use your product. In what situation would they use it? And who's your target audience?

If you get stuck at this stage, try to identify your personal challenges. Is there anything you would be able to do better if you had the right tool?

2 EVALUATE YOUR COMPETITORS

Once you've identified the main goal of your product it's time to conduct a competitor analysis to see what your competition looks like.

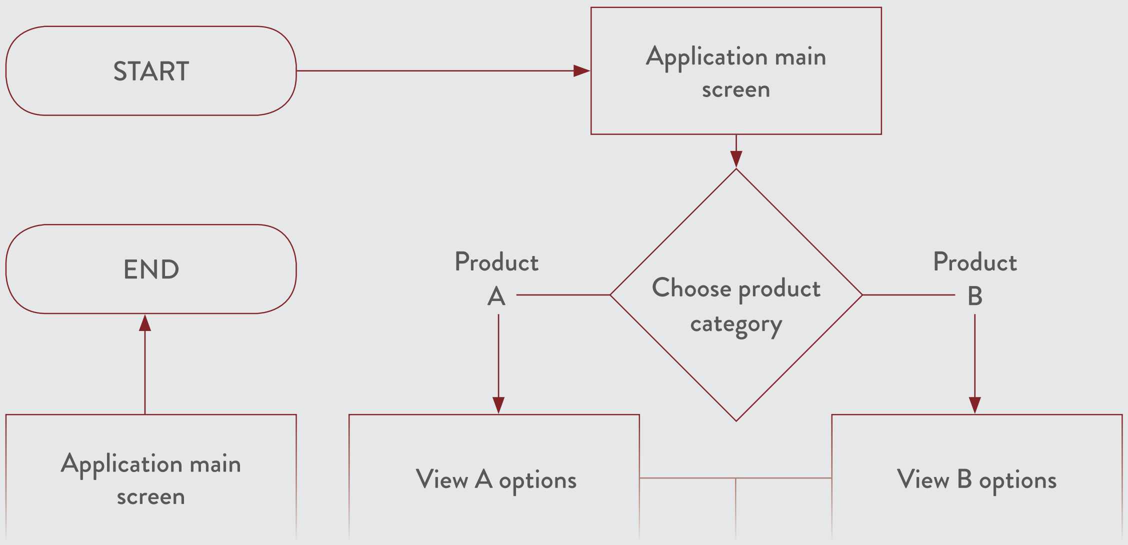
Start-ups often don't take this step into consideration. They simply move forward with their development in blind faith – which is never a good idea. Your competitors will have made mistakes that you can learn from, and these are valuable shortcuts. So take the time to look at your competitor's websites and apps.

3 DEFINE THE MOST BASIC USER FLOW POSSIBLE

A user flow consists of a number of steps that need to be performed in order for a task to be completed. In order to define your user flow, you need to define process stages. Simply put, define the steps needed to reach your product's primary goal.

Don't get involved with your product's features at this point. This will come later. Take a more basic approach to define your user flow.

What is the solution your user will have when they use your product?



4 STRIP BACK YOUR FEATURES

Now that you've created your user flow, you can start creating a more detailed list of features for each particular stage of the flow.

Once you've laid out your features for each stage, you will need to prioritise them. What is the most important action you need your users to complete to help them solve their problem?

This will be your main feature and solve one or two important customer problems. Think about each stage of your product and how a list of features might fit into each stage.

Now that all of your features are listed out, try to weed out the least important features. It might help to categorize your features from top to bottom by priority with 'must-haves,' 'should haves,' and anything that doesn't fit those two criteria can go straight in to the bin at this stage.

Once you've identified all of the must-have features for each stage, it's time to define their scope for the first version of your product. During this stage, a lot of start-ups will begin creating a prototype.

Read more about using wireframes and prototypes on our blog

5 **BUILD, TEST, AND LEARN**

Here is where you'll move towards the development and testing stage of your product.

Continuous testing helps you to improve the quality of the product before it's been released.

Once your product has reached its alpha stage you will want to release it to a small group of people or real users through your first rounds of user testing.

It's nearly always the case that no software product survives first contact with its users. You will need to make changes and enhancements to your initial concept, and that's fine!

The testing stages of your product are crucial for you to understand what features your product lacks and what features your product can do without. Your users are the best people to define this.

From here, you can continue to improve your product by testing, learning from your consumers, building, testing and learning again.

CONCLUSION

If there's one takeaway point from creating a minimum viable product, it's that you should keep things as simple as possible.

Remember to ask yourself - what is the core value of your product?

Test, test, test. And test some more. That way, you'll give yourself room to fail but do it quickly and frequently.

Regardless of how you move forward, I wish you all the best with your new product!

SIMON SWORDS

Managing Director, Atlas Computer Systems Limited

simon.swords@atlascode.com

ATLAS
_CLEVER SOFTWARE

BESPOKE SOFTWARE THAT HELPS YOUR BUSINESS GROW