Running Lean

A systematic process for iterating your web application from Plan A to a plan that works

by Ash Maurya

"Customer Development" is a term coined by Steve Blank and is used to describe the parallel process of building a continuous feedback loop with customers throughout the product development cycle as defined in his book: "The Four Steps to the Epiphany".

"Lean Startup" is a term trademarked by Eric Ries and represents a synthesis of Customer Development, Agile software development methodologies, and Lean practices.

"Lean Canvas" is a term trademarked by Ash Maurya and is used to describe an adaptation to the business model canvas released under the Creative Commons license by Alex Osterwalder.

Formatting inspired by 37signals and Venture Hacks.

First edition

CONTENTS

INTRODUCTION

What is Running Lean?

About the Author

Disclaimers

What is Running Lean?

We live in an age of unparalleled opportunity for innovation. With the advent of the Internet, Cloud Computing, and Open Source software, the cost of building software is at an all time low. Yet, the odds of building successful products haven't improved much:

9 out of 10 startups still fail.

And of those that succeed, more than two-thirds report changing their plans drastically along the way. What separates successful startups is not necessarily starting with a better initial plan (Plan A), but **finding a plan that works** before running out of resources.

Up until now, finding this better plan was based more on gut, intuition, and luck. There has been no systematic process for rigorously stress testing a Plan A.

That is what Running Lean is about:

Running Lean is a systematic process for iterating from Plan A to a plan that works.

Why are startups hard?

First, there is a misconception around how successful products get built. The media loves stories of visionaries that see the future and chart a perfect course to intersect it. The reality, however, rarely plays out quite as simply. Even the unveiling of the visionary computer, the iPad, in Steve Jobs words were years in the making, built on several incremental innovations (and failures) of software and hardware.

Second, the classic product-centric approach front-loads some customer involvement during the requirements gathering phase but leaves most of the customer validation until after the software is released. There is a large "middle" when the startup disengages from customers for weeks or months while they build and test out their solution. During this time, it's quite possible for the startup to either build too much or be led astray from building what customers want. This is the fundamental dilemma described by Steve Blank in "The Four Steps to the Epiphany" in which he offers a process for building a continuous customer feedback loop throughout the product development cycle that he terms: "Customer Development".

And finally, even though customers hold all the answers, you simply cannot ask them what they want. Given the right context, customers can clearly articulate their problems but it's your job as the visionary entrepreneur to come up with solutions.

"If I had asked people what they wanted, they would have said faster horses."

- Henry Ford

Is there a better way?

Running Lean provides a better way to build web applications in the face of extreme uncertainty.

> Running Lean is about speed.
>
> Running Lean is about testing a vision by measuring how customers behave.
>
> Running Lean is about engaging customers throughout the product development cycle.
>
> Running Lean tackles both product and market validation in parallel using short iterations.
>
> Running Lean is a disciplined and rigorous process.

But most importantly, Running Lean is about focusing on the right stuff and reducing waste, a philosophy which can be summed up as:

> *"RIGHT ACTION, RIGHT TIME"*
>
> - Bijoy Goswami
> Founder, Bootstrap Austin

Running Lean is a workflow for building web based software that I refined based on my own experiences of rigorously applying and testing **Customer Development, Lean Startup, and Bootstrapping** techniques to my products.

“Customer Development” is a term coined by Steve Blank and is used to describe the parallel process of building a continuous feedback loop with customers throughout the product development cycle as defined in his book: “The Four Steps to the Epiphany”.

“Lean Startup” is a term trademarked by Eric Ries and represents a synthesis of Customer Development, Agile software development methodologies, and Lean (as in the Toyota Production System) practices.

“Bootstrapping” is more commonly understood as a collection of techniques used to minimize the amount of external debt or funding needed from banks or investors. People too often confuse bootstrapping with self funding. I subscribe to a stricter definition of Bootstrapping, which is **funding with customer revenues**.

In this book, you'll learn:

How to first find a problem worth solving, before defining a solution

How to find early customers

When is the ideal time to raise funding

How to test pricing

How to decide what goes into Release 1.0

How to build and measure what customers want

How to maximize for speed, learning and focus

What is Product/Market Fit?

How to iterate to Product/Market fit

Is this book for you?

If you are an entrepreneur considering building a new web application, or already have one, and want to raise your odds for building a successful product, this book is for you.

While several of the techniques described here pertain to web based software, the core principles behind Running Lean are applicable to many other types of projects such as writing a book, starting a blog, even finding a co-founder.

About Me

I bootstrapped my last company, WiredReach, 7 years ago and have launched 4 products and 1 peer-to-web application framework. Throughout this time, I've been in search for a better way to build products: I've launched products in stealth, attempted building a platform, dabbled with open sourcing, applied bootstrapping techniques, built too many features, embraced fewer features, and tried release-early release-often.

The first realization early on was that building in stealth is a really bad idea. There is this fear, especially common among first-time entrepreneurs, that their great idea will be stolen by someone else. The truth is twofold: First, most people are not capable of visualizing the potential of an idea at such an early stage, and second (and more importantly) that **they won't care**.

The second realization was that while listening to customers is important, **you have to know how**. I used a "release-early, release-often" methodology for one of my products, BoxCloud, and launched a fairly minimal file-sharing product built on a new peer-to-web model we had developed. After we launched, we got covered by a few prominent blogs and dumped some serious cash into advertising on the DECK network (primarily targeted at designers and developers).

We started getting a lot of feedback but it was all over the place. We didn't have a clear definition of our target customer and didn't know how to prioritize the feedback. We started listening to the most popular (vocal) requests and ended up with a bloated application and lots of one-time use features.

At around that time, I ran into Steve Blank's lectures on "Customer Development". I had dreamt the big vision, rationalized it in my head, built it and refined it the long, hard way. I knew customers held the answers but didn't know when or how to fully engage them. That's exactly what Customer Development addresses. I was sold.

Why this book?

I was determined to apply these techniques to my next product, CloudFire, but ran into some tactical challenges when trying to take these concepts to practice. For one, Steve Blank's book was written for a specific type of business - Enterprise software (not Web software), which made it hard to carry over some of the tactics. While Eric Ries was sharing his retrospective lessons learned from working at IMVU, what you saw was a fully realized Lean Startup machine which was at times daunting.

I had more questions than answers and decided the best way to answer them was not just to try and adapt these techniques in isolation but to openly test them and blog about it. Writing not only crystallized my own thinking but also forced me to research and really grok a topic.

Along the way, I've made new friends with people that have greatly influenced and continue to refine my thinking: Eric Ries, Steve Blank, Nivi Babak, Dave McClure, Sean Ellis, Hiten Shah, Sean Murphy, Jason Cohen, and Joshua Baer.

I am thankful to the thousands of readers that subscribed to my blog, left comments week after week, and sent me notes of encouragement to keep on writing. This book was really "pulled" out of me by my readers.

Field tested

As a way to test the content for this book, I started speaking and teaching "Running Lean" workshops. I have shared the Running Lean workflow with hundreds of startups and worked closely with many of them to test and refine it.

While the blog is a near-realtime account of my lessons learned, the book benefits from retrospective learning, reordering, and refining of steps for a more optimal workflow.

I am applying this new workflow to my next startup, **USERcycle**, which is also a byproduct of my blogging and learning over the last year. As of this writing, I have sold my last company, WiredReach, and am in the process of building and launching two products:

Lean Canvas is a business model validation tool. It's a companion tool to this book that simplifies how you document business models, measure progress, and communicate learning with your internal and external stakeholders.

USERcycle is lifecycle marketing software that helps companies convert their users into passionate customers. Passionate customers come back and use your product, tell others about your product, and pay for your product (or get you paid).

Disclaimers

I have a rule of only writing about things that I have learned first-hand through building and running my products. I don't believe in extrapolating what worked for my type of products to other models. So for instance, I've never built a multi-sided (marketplace) product like eBay, or one that solely relied on huge network effects like Facebook or twitter. Even though I definitely have ideas on how I'd apply this process there and am helping other startups do so, without first-hand experiential learning, I will not present them here.

For these reasons, it is important for me to lay out the characteristics of the kinds of products I've built and what I believe.

I've built:

- Software as a Service (SaaS) applications
- Web-based and desktop-based software
- B2B and B2C applications
- Freemium and Free Trial products

I believe:

- If you're ever going to charge for your service, charge from day one
- You need to build and test a path to customers from day one
- Practice trumps theory

A Methodology is Just That

Lastly, no methodology can guarantee success. There are no silver bullets. But a good methodology provides a feedback loop for continuous improvement and learning.

While Running Lean does not guarantee success, it helps raise your odds for building a successful product.

That is the promise of this book.

CHAPTER 1

ROADMAP

Getting to Product/Market Fit is the first significant milestone of a startup. You get there by first documenting your Plan A and then systematically testing and refining it through well defined experiments that maximize for speed, learning, and focus.

Document Your Plan A

There is an "I" in vision

"All men dream: but not equally. Those that dream by night in the dusty recesses of their minds awake to find that is was vanity: but the dreamers of day are dangerous men, for they may act their dreams with open eyes, to make it possible."

- T.E. Lawrence
Lawrence of Arabia

Everyone gets hit by ideas when they least expect them (in the shower, while driving, etc.). Most people ignore them - entrepreneurs choose to act on them.

While passion and determination are essential attributes needed to drive a vision to its full potential, left unchecked they can also turn the journey into a faith-based one driven by dogma.

Reasonably smart people can rationalize anything but entrepreneurs are especially gifted at this.

Most entrepreneurs start with a strong initial vision and a Plan A for realizing that vision. Unfortunately, **most Plan As don't work**.

While a strong vision is required to create mantra and make meaning, a Lean Startup strives to uphold a strong vision with facts not faith. It is important to accept that your initial vision is built largely on untested hypotheses. Running Lean helps you systematically test and refine that initial vision.

Capture your business model hypotheses

The first step is writing down your initial vision and sharing it with at least one other person. Too many founders carry their hypotheses in their heads alone which, while the fastest way to iterate, only helps to further their own "reality distortion fields".

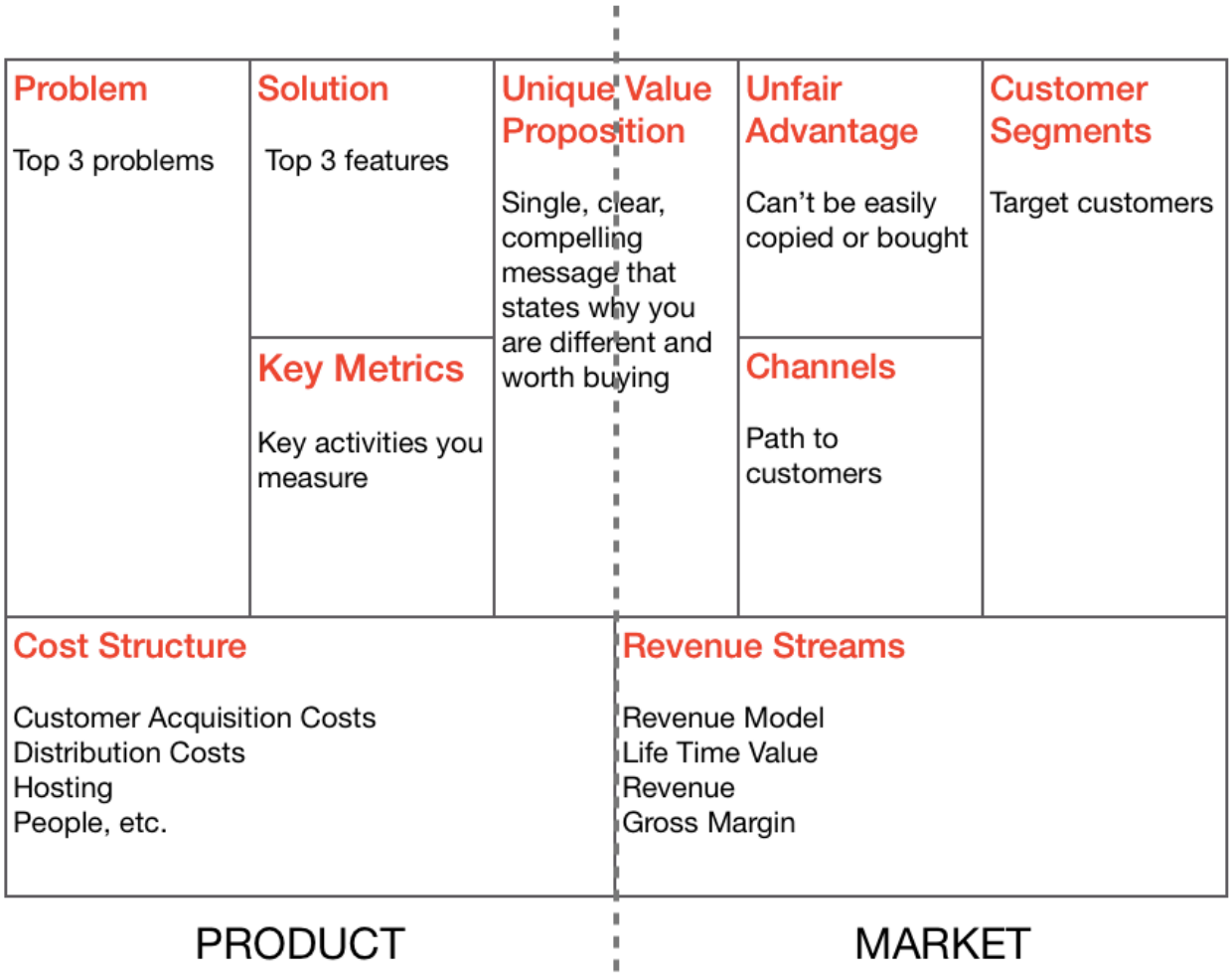
There are several techniques for capturing business model hypotheses. Business Plans have traditionally been used for this purpose. While the intent of business plan writing is sound, taking several weeks or months to write a 60-page business plan largely built on untested hypotheses (guesses) is a form of waste.

Business Plan: A document investors make you write that they don't read.
- Steve Blank

More importantly, since most Plan As are likely to be proven wrong anyway, you need something less static and rigid than a business plan.

Steve Blank describes another way to capture hypotheses in his book using a set of detailed worksheets which is how I got started. At first, I refined his worksheets for my use but I still struggled with keeping them up-to-date over time. I have since then settled on a single page "Lean Canvas" format for visualizing and testing hypotheses.

Lean Canvas is my adaptation of Alex Osterwalder's Business Model Canvas which he describes in his book: "Business Model Generation".



If you have ever written a business plan or created a slide deck for investors, you'll immediately recognize most of the building blocks on the canvas.

I particularly like the 1-page canvas format because it is:

Fast
Compared to writing a business plan which can take several weeks or months, you can outline multiple business models on a canvas in one afternoon.

Concise
The canvas forces you to pick your words carefully and get to the point. This is great practice for distilling the essence of your product. You have 30 seconds to grab the attention of an investor over a metaphorical elevator ride, and 8 seconds to grab the attention of a customer on your landing page.

Portable
A single page business model is much easier to share with others which means it will be read by more people and probably also more frequently updated.

I'll cover detailed steps to creating a Lean Canvas in Part 1 of this book.

Systematically Test Your Plan

With your Plan A documented, the next step is **systematically** testing that plan through a set of well defined experiments. Tackling the entire plan all at once is not only overwhelming but leads to waste.

Waste is any human activity which absorbs resources but creates no value.

- Womak and Jones
Lean Thinking

For instance, too many startups expend effort prematurely optimizing their user acquisition channels or strategic partnerships before they even have a product that works.

Startups are inherently chaotic but at any given point in time there are only a few actions that really matter. Running Lean is about focusing on those and ignoring the rest - **Right Action, Right Time**.

The right actions are dictated by the stage of your startup:

Stage 1: Problem/Solution Fit

Key Question: Do I have a problem worth solving?

Before investing months or years of effort towards building a product, the first step is determining if this product is **something worth doing**. You do this by decoupling the problem from the solution and testing each through customer interviews - a process Steve Blank calls “Customer Discovery”.

Testing the problem this way lets you validate whether you have a “problem worth solving” before investing effort building out a solution. From there you then derive the minimum feature set to addresses the right set of problems - the **Minimum Viable Product** (MVP). We’ll cover the detailed steps for validating Problem/Solution Fit in Part 1 of this book.

Stage 2: Product/Market Fit

Key Question: Have I built something people want?

Once you have a problem worth solving and your Minimum Viable Product built, you can then start the process of learning from customers and testing how well your solution solves the problem. Lean Startups are all about maximizing this type of learning which we’ll cover in Part 2 of this book.

Achieving traction or Product/Market Fit is the first significant milestone for a startup. At this stage you have a plan that is starting to work - you are signing up customers, retaining them, and getting paid. We’ll cover how you measure and iterate towards Product/Market Fit in Part 3 of this book.

Stage 3: Scale

Key Question: How do I accelerate growth?

After Product/Market Fit some level of success is almost always guaranteed. Your focus then shifts towards accelerating your plan for scale.

Pivot before Product/Market Fit, Optimize after

Another way to delineate the stages of a startup is before and after Product/Market Fit.

Before Product/Market Fit, the focus of the startup centers around **learning and pivots**. After Product/Market fit, the focus shifts towards **growth and optimization**.

"Pivot" is a term used by Eric Ries to describe a change in direction of a startup while staying grounded in learning. The best way to differentiate pivots from optimizations is that pivots are about **finding a plan that works**, while optimizations are about **accelerating that plan**.

In a pivot experiment, you attempt to **validate** parts of the business model hypotheses in order to find a plan that works. In an optimization experiment, you attempt to **refine** parts of the business model hypotheses in order to accelerate a working plan. The goal of the first is a course correction (or a pivot). The goal of the second is efficiency (or scale).

This may sound like a subtle distinction but it has a significant impact on both strategy and tactical execution. Before Product/Market fit a startup needs to be architected for maximizing learning.

You stand to learn the most when the probability of the expected outcome is fifty percent i.e. you don't know what to expect.

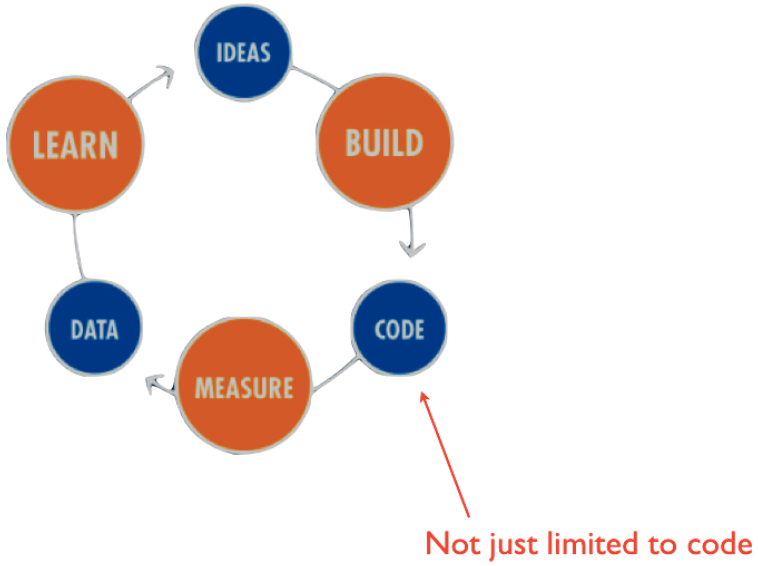
In order to maximize learning, you have to **pick bold outcomes** versus chase incremental improvements. So, rather than changing the color of your call to action button, change your entire landing page. Rather than tweaking your unique value proposition for a single customer segment, experiment with different unique value propositions for different customer segments.

We'll visit many other examples later in the book on how you purposely architect for learning over optimization.

Build a Validated Learning Loop

In this section, I'll describe how you systematically test your plan through **experiments**.

Eric Ries succinctly describes the fundamental validated learning loop in a Lean Startup as Build/Measure/Learn:



This validated learning loop begins in the **"build stage"** with a set of ideas or hypotheses that are used to create some artifact (mockups, code, landing page, etc.) for the purpose of testing a hypothesis. Responses from customers are collected in the "**measure stage"** and the resulting data is used during the **"learn stage"** to validate or refute a hypothesis which in turn drives the next set of actions.

A cycle around the validated learning loop is called an experiment.

The ability to correctly define and run experiments is a basic requirement and one that I've seen a number of startups struggle with in the workshops I teach.

Some of this difficulty is understandable as Lean Startups attempt to apply the "Scientific Method" to a startup which isn't an exact science. But without a certain level of rigor and objectivity, there is a danger of creating a *data-driven pattern-seeking organization* that might actually be just as bad or even worse at furthering "rationalized dogma".

Here are some ground rules we'll use throughout the book:

1. Formulate testable hypotheses
What most people write down as business model hypotheses are really not yet in a form that make them testable. The goal should be clearly defining the conditions under which a hypothesis can be absolutely proved or disproved. Otherwise, you easily fall into the trap of accumulating just enough evidence to convince yourself that your hypothesis is correct.

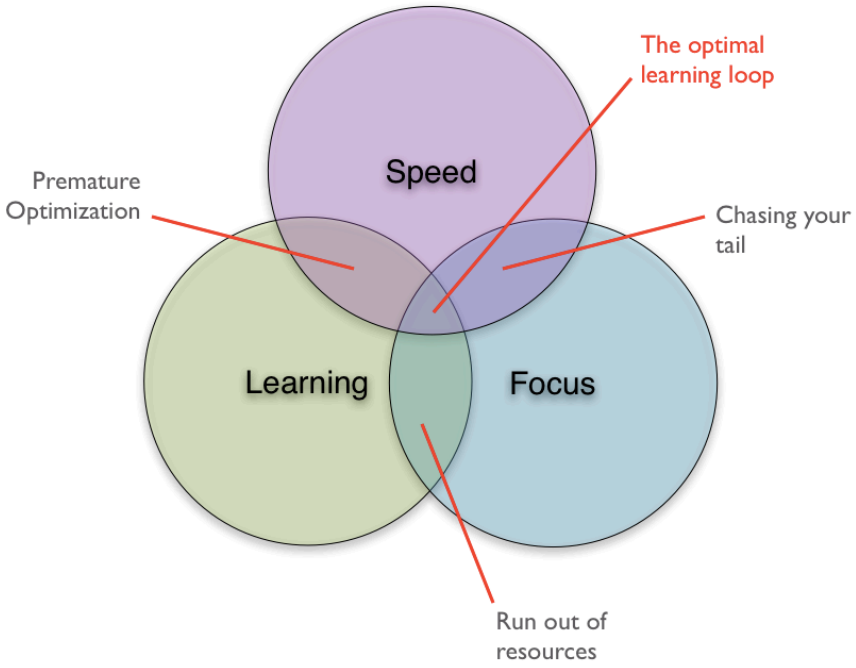
Here is an example:

Not Measurable: Being known as an "expert" will drive early adopters.
Specific and Testable: Blog post will drive 100 sign-ups.

The first statement is a hypothesis that cannot be proved wrong because the expected outcome of driving early adopters is not measurable. Specifically, it is not clear how many early adopters are needed to prove this hypothesis true - 1 or 100 or 1000? The second statement not only has a specific outcome but it's also based on a repeatable action which makes it testable.

2. Maximize for Speed, Learning, and Focus

While Eric Ries stresses the importance of speed through the validated learning loop, something that doesn't get as much attention is focus. This is best epitomized by Bijoy Goswami as: Right Action, Right Time.



Speed and Learning alone can lead to pre-mature optimization which is a form of waste. You need all three – **Speed, Learning, and Focus**, to build an optimal validated learning loop.

3. Validate Qualitatively, Verify Quantitatively

Before Product/Market fit, you typically do not have enough traffic to afford waiting for statistically significant results. The good news is that if you are maximizing for learning and picking bold outcomes, that naturally works to your advantage.

Your initial goal is getting a strong signal (positive or negative) which typically doesn't require a large sample size. You might be able to do this with *as few as five customer interviews*[1].

A strong negative signal indicates that your bold hypothesis most likely won't work and lets you quickly abandon or refine it. On the other hand, a strong positive signal also doesn't necessarily mean it will scale up to statistical significance either. But it gives you permission to move forward on the hypothesis until it can be verified later through quantitative data.

Validating hypotheses this way - first qualitatively, then quantitatively, is another key principle from Running Lean that we'll see applied in various stages throughout the book.

4. Create accessible dashboards

Testing hypotheses can be scary for founders. This is understandable as startup founders pour their blood, sweat, and tears into their work. They have a lot riding on the outcome of their efforts and don't like to be proven wrong. But without a level of transparency and objectivity, there is a danger of running your startup primarily on faith. It is imperative not only to share your hypotheses company-wide but also the results of your experiments.

"A business should be run like an aquarium, where everybody can see what's going on."

- Jack Stack
The Great Game of Business

[1] This number comes from usability testing research (via Jakob Nielsen/Steve Krug) that shows how five testers are enough to uncover 85 percent of the problems. We'll also see some specific examples later in the book where I've been able to verify this claim.

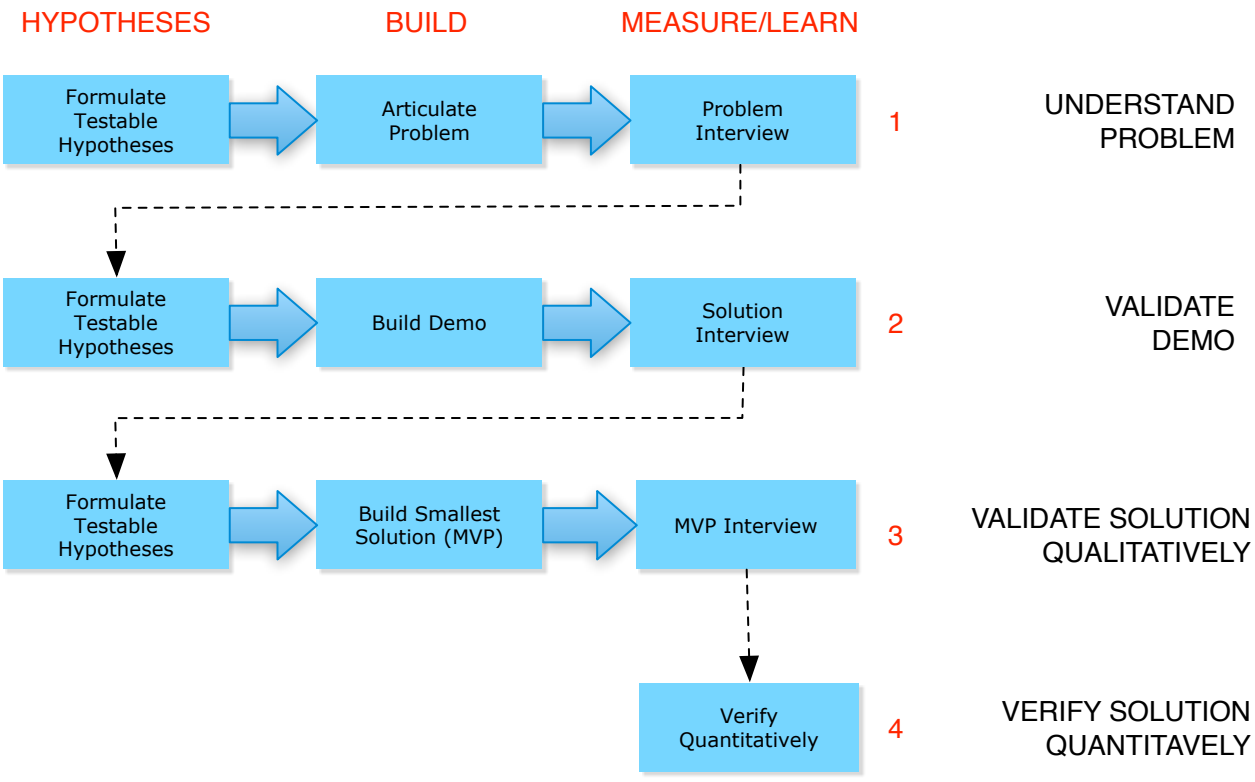
5. **Communicate learning early and often**
Company-wide dashboards are great for on-the-ground tactical analysis, but it is equally important to report on your learning at a strategic level. A good way to do this is by periodically communicating the lessons learned from your last batch of experiments - weekly with your internal team and at least once a month with your external advisors and investors. This lets you pause, reflect on your findings as a team, and better plan out the next set of activities i.e. hypotheses to test.

Communicating progress in this way lets you stay grounded in learning while constantly iterating towards a plan that works.

Iterate Your Way to Product/Market Fit

While an experiment helps you validate a specific business model hypothesis, an **iteration** strings multiple experiments together into a workflow for validating a "product" from idea to launch to Product/Market Fit.

The basic iteration meta pattern we'll use throughout this book is shown below:



If you look closely, you'll see this workflow is made up of four Build/Measure/Learn loops - three of them are qualitative and one is quantitative.

The terrain before Product/Market Fit is riddled with qualitative learning.

Here's a high-level overview of the iteration meta pattern:

1. Understand Problem
Your goal is first determining if you have a problem worth solving and second understanding how customers solve this problem today i.e. what are the existing alternatives.

Before you can build the right solution, you have to understand the problem first.

2. Validate Demo
Armed with an understanding of the problem, you are then in the best position to define a possible solution. But rather than jumping to implement the actual solution, you need to test your possible solution qualitatively with customers using a demo that helps them visualize the solution.

Customers are better at articulating problems than visualizing solutions.

3. Validate Qualitatively
You use the feedback from the demo to distill down your product to it's essence (MVP) which you then build and test qualitatively with customers.

4. Verify Quantitatively
While getting a strong qualitative signal from customers grants you permission to move forward, it does not guarantee scalability. You still need to verify that using actionable macro metrics.

A good way to see this iteration meta-pattern in action is to consider a non-software related case study.

Case Study- How I wrote this book

Customer Pull
Writing a book was never in my plans. I started my blog simply because I had more questions than answers. Along the way, a few of my blog readers started suggesting that I turn my blog posts into a book. I knew writing a book (even from blog posts) is a massive undertaking so while I was flattered by the requests, **I did nothing at first**. After about a dozen such requests, I decided to explore further.

Understand Problem
I called these readers up and asked them why they wanted me to write a book. Specifically I asked what would be different about this book from what was already on my blog, Eric Ries's blog, or Steve Blank's book? (**existing alternatives**)

From these interviews, I learned that, like me, they too were struggling taking Customer Development and Lean Startup techniques to practice (**problem**) and viewed my blog posts as a "step-by-step" guide for applying these techniques from the ground up.

While Eric Ries succinctly captures the validated learning loop with Build/Measure/Learn, what people wanted wanted to know was how to string one or more of these learning loops into a workflow for taking a product from idea to launch to Product/Market Fit.

Define Solution
With that knowledge, I spent a day **building a demo**. It was a landing page with a table of contents, a title, a blank book cover image, and a price.

I called up the same readers and asked them if I were to write this book would they buy it? Their feedback gave me a strong signal to move forward and helped me **define the solution**. I ended up refining the original title, price, and table of contents.

While encouraging, I still didn't want to write a book for just a dozen or so readers so I announced the book on my blog and others helped spread the message (**channel testing**). Once I hit 1,000 potential buyers, this became a **problem worth solving** for me. My rationale for this thinking was to at least cover my costs for writing this book using a simple back-of-the-envelope calculation.

Validate Qualitatively

Writing a whole book was still a massive undertaking so I needed to build the smallest thing possible to start learning from customers (**a minimum viable product**).

I took the table of contents and turned it into a slide deck with the same outline and a few bullet points under each topic. I announced a free Running Lean workshop and got 30 people interested. I launched the first workshop with 10 people who helped me test and refine the content. Based on the success of the first workshop, I ran more workshops and started **charging** for them. With each workshop, I continually tweaked the slide deck content for better flow.

Once I understood the solution, I started writing. Here again, instead of writing the whole book in isolation, I contacted my potential prospects from the teaser page (some of who were growing impatient) and told them I'd be writing and releasing the book **iteratively**: Rather than waiting six months to get the book, if they pre-ordered the book, they would get 2 chapters of the book every 2 weeks in a PDF format.

Enough people agreed to this arrangement which allowed me to further test and refine the content iteratively with **my early adopters**. The early adopters were driven by the content alone and didn't care about typos or an iPad/Kindle version. Others chose to wait for the "finished product".

Verify Quantitatively

As of this writing, the book is now "content-complete" and almost ready for a full launch.

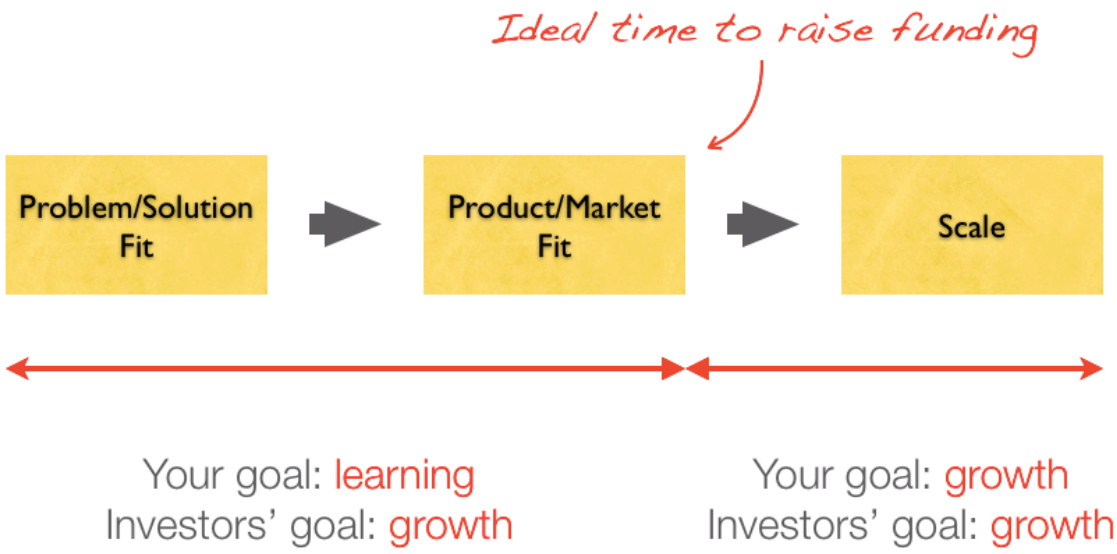
Up until now, my focus has been solely on content. I have deferred things like designing a book cover, testing the book sub-title, or researching print/eBook options all of which I am doing now (**Right Action, Right Time**).

While I've always been prepared to self-publish this book, I've been contacted by two major publishers that are currently reviewing the manuscript. I asked them if my model for writing and selling the book so far would be a deal-breaker. On the contrary, they wished more authors wrote their books this way.

By selling the book on my own and demonstrating **early traction**, I was proving there was a market for this book and helping mitigate market risk for a publisher. As with building a web-based product, the ideal time to attract external resources is after Product/Market Fit which ironically may or may not be the right action at that time. The final decision to go the publisher route or self-publish is still being determined – economics, reach, and time to market are all factors.

What About Funding?

It's funny to note how the 37signals folks went from "Outside money is Plan B" to "Outside money is Plan Z" between their last two books. Once you're profitable, it's easy to make such a declaration but there are certainly better times than others to consider external funding.



The ideal time to raise funding is **after Product/Market Fit** because at that time both you and your investors have aligned goals - to scale the business.

Traction is a measure of your product's engagement with its market. Investors care about traction over everything else.

- Nivi and Naval
Venture Hacks

A lot of (especially first-time) entrepreneurs feel that step one is writing a business plan and getting funded. Taking several months to write and then pitch a 60-page business plan document to investors is not the best use of time for a startup. Especially since all you have at that point is a vision and a set of untested guesses. Selling this to investors without any level of validation is a form of waste.

I've bootstrapped my company for the last seven years and while not the same thing, Bootstrapping and Lean Startups are quite complementary. Both cover techniques for building **low-burn startups** by eliminating waste through the **maximization of existing resources** before expending effort on the acquisition of new or external resources.

Bootstrapping + Lean Startup = Low Burn Startup

(See the Appendix: How to Build a Low-burn Startup)

The Road Ahead

The rest of the book will cover a detailed workflow for taking a product from idea to launch to Product/Market Fit.

This workflow synthesizes the ideas from bootstrapping, lean startups, and customer development presented earlier and is organized with the goal of **accelerating learning and reducing waste**.

Part 1: Problem/Solution Fit
Do you have a problem worth solving?

Part 2: Product/Launch Fit
Are you ready to learn from customers?

Part 3: Product/Market Fit
Have you built something customers want?

Each section ends with gating criteria that help you decide if you're ready to move on the next section. Let's begin.

PART 1: PROBLEM/SOLUTION FIT

SUMMARY

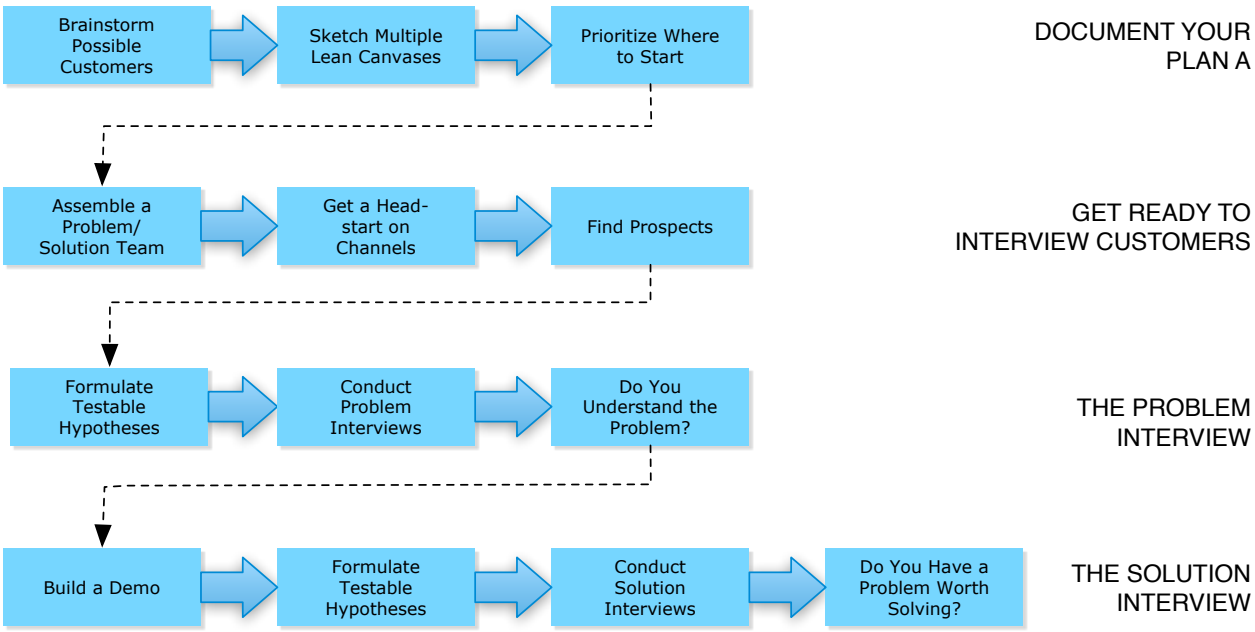
Problem/Solution Fit, in one page.

A problem worth solving boils down to three questions:

1. *Is it something customers want? (must-have)*
2. *Can it be solved? (feasible)*
3. *Will they pay for it? If not, who will? (viable)*

A basic Lean Startup technique is doing the smallest thing possible to learn from customers. We apply this technique here to validate Problem/Solution Fit through a pair of **carefully scripted customer interviews** before investing the effort in defining and building a solution.

The process that follows is a refinement of the "Customer Discovery" process Steve Blank describes in his book: "The Four Steps to Epiphany".



CHAPTER 2

CREATE YOUR LEAN CANVAS

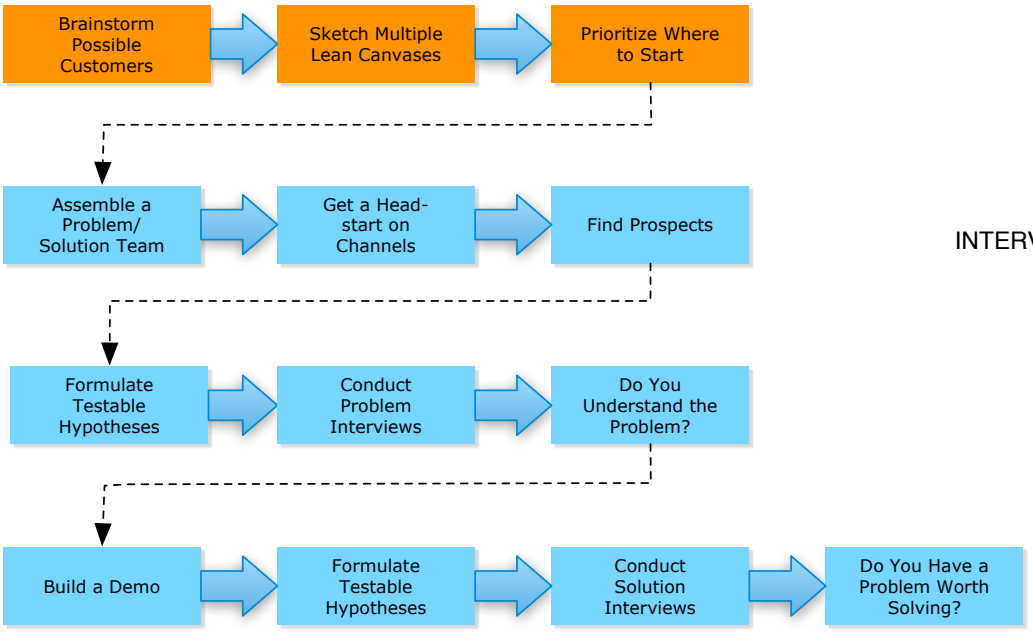
Capture your business model in a portable 1-page diagram. The Lean Canvas is the perfect format for brainstorming possible business models, prioritizing where to start, and tracking ongoing learning.

The best way to illustrate the use of the canvas is through an example. I'll describe the thought process that went into building a Lean Canvas for the companion business model hypotheses testing tool - also called Lean Canvas.

Brainstorm Possible Customers

Sketch Multiple Lean Canvases

Prioritize Where to Start


Brainstorm Possible Customers
Sketch Multiple Lean Canvases
Prioritize Where to Start
DOCUMENT YOUR PLAN A
Assemble a Problem/ Solution Team
Get a Head-start on Channels
Find Prospects
GET READY TO INTERVIEW CUSTOMERS
Formulate Testable Hypotheses
Conduct Problem Interviews
Do You Understand the Problem?
THE PROBLEM INTERVIEW
Build a Demo
Formulate Testable Hypotheses
Conduct Solution Interviews
Do You Have a Problem Worth Solving?
THE SOLUTION INTERVIEW

Brainstorm Possible Customers

You most likely already have an inkling of the problem, solution, and customer in mind. Start by brainstorming the list of possible customers you envision using your product.

1. Distinguish between customers and users
If you have multiple user roles in your product, identify customers.

A customer is a someone that pays for your product.

2. Split broad customer segments into smaller ones
I've worked with startups that felt the problems they are solving are so universal, they apply to everyone.

You can't effectively build, design, and position a product for everyone.

While you might be aiming to build a mainstream product, you need to start with a specific customer in mind. Even Facebook, with it's now 500 million+ users started with a specific user in mind - Harvard college students.

3. Sketch a Lean Canvas for each customer segment
As you'll find shortly, the elements of your business model can and will vary greatly by customer segment. I recommend starting with the top 2-3 customer segments you feel you understand the best or find most promising.

Case-study: Lean Canvas

Background
In the course of applying Customer Development and Lean Startup principles to my products, I inevitably needed to document my business model hypotheses. I started with the worksheets at the end of Steve Blank's book: "The Four Steps to the Epiphany".

While this was a great exercise, the output was a collection of documents and spreadsheets that become hard to manage and share with other team members over time. I then ran into Alex Osterwalder's work on Business Model Canvas. While I really liked the portability and clarity of a 1-page business model, I found some of his canvas elements too general and started toying with an adaptation that eventually became the Lean Canvas. The scope of the problems I was addressing also grew from just documenting hypotheses, to measuring and communicating progress over time.

When I started, I envisioned the Lean Canvas being a replacement for worksheets and business plans which led to the following "possible customers" brainstorm list:

Really broad category: Anyone that uses a business plan today

More Specific Possible Customers:
1. Startup founders (bootstrapped, funded)
2. Startup accelerators
3. Investors (Angels, VC)
4. Large companies

Since this was a "scratch my own itch" kind of problem, I decided to build my first Lean Canvas for **startup founders**.

Sketch Multiple Lean Canvases

In this section, I'll outline the process for sketching a Lean Canvas.

1. Sketch a canvas in one sitting
While a business plan can take weeks or months to write, your initial canvas should be sketched quickly.

2. It's okay to leave sections blank
Rather than trying to research or debate the "right" answers, put something down quickly or leave it blank and come back to it later. Some elements like "Unfair Advantage" take time to figure out. The canvas is meant to be an organic document that evolves over time and it's okay to say "I don't know".

3. Think in the present
Business plans try too hard to predict the future which is impossible. Instead, write your canvas with a "getting things done" attitude. Based on your current stage and what you know right now, what are the next set of hypotheses you need to test to move your product forward?

4. Use a customer-centric approach
While Alex Osterwalder discusses several alternative approaches to sketching an initial canvas in his book, I prefer using a customer-centric approach. I start with the Customer Segment and follow a prescribed order to filling out a canvas:

<table>
<tr>
<td rowspan="2">Problem

Top 3 problems

1</td>
<td>Solution

Top 3 features

3</td>
<td rowspan="2">Unique Value Proposition

Single, clear, compelling message that states why you are different and worth buying

2</td>
<td>Unfair Advantage

Can’t be easily copied or bought

7</td>
<td rowspan="2">Customer Segments

Target customers

1</td>
</tr>
<tr>
<td>Key Metrics

Key activities you measure

6</td>
<td>Channels

Path to customers

4</td>
</tr>
<tr>
<td colspan="3">Cost Structure

Customer Acquisition Costs
Distribution Costs
Hosting
People, etc.

5</td>
<td colspan="2">Revenue Streams

Revenue Model
Life Time Value
Revenue
Gross Margin

5</td>
</tr>
</table>

Problem and Customer Segments

I find that the "Problem-Customer" pair usually drives the rest of the canvas which is why I tackle them together:

1. List top 3 problems
For the customer segment you are working with, describe the top 1-3 problems they need solved.

2. List existing alternatives
Then document how you think your early adopters address these problems today. Unless you are solving a brand new problem (unlikely), most problems have existing solutions. Many times these may not be a readily obvious competitor.

As an example, the biggest alternative to most online collaboration tools is not another collaboration tool, but email. Doing nothing could also be a viable alternative for a customer if the pain is not acute enough.

3. Identify other user roles
Next identify any other user roles that will interact with this customer.

Examples:
1. In a blogging platform, the customer is the blog author while the user is a reader.
2. In a photo sharing service, the customer is the sharer, while users are viewers (family and friends).

4. Hone in on possible early adopters

With these problems in mind, get more specific on the customer segment. Narrow down the distinguishing characteristics of your prototypical customer.

Your objective is to define an early adopter - not a mainstream customer.

As an example, my last product, CloudFire, was a photo and video sharing service targeted at parents. Further refinement got me to define my early adopters as “first-time moms with kids under the age of three”.

Case Study: Lean Canvas - Problem and Customer Segments

Customer Segment: Startup Founder

1. Top 3 problems

Problem 1: Business Models need to be more portable

Business Plans are too static and don’t get read or updated enough. A lot of entrepreneurs skip this step altogether which is NOT the solution.

Problem 2: Measuring progress is hard work

Customer Discovery is a qualitative process and requires a lot of work (documenting interviews, aggregating results, etc.) to yield actionable insights.

Problem 3: Communicating learning is critical

The biggest mind shift when practicing a Lean Startup methodology is being objective and instilling a culture of continuous learning and improvement by holding ourselves accountable to internal and external stakeholders.

2. Existing Alternatives

Intuition, business plans, worksheets/spreadsheets.

3. User Roles

Creators (Startup Founders):

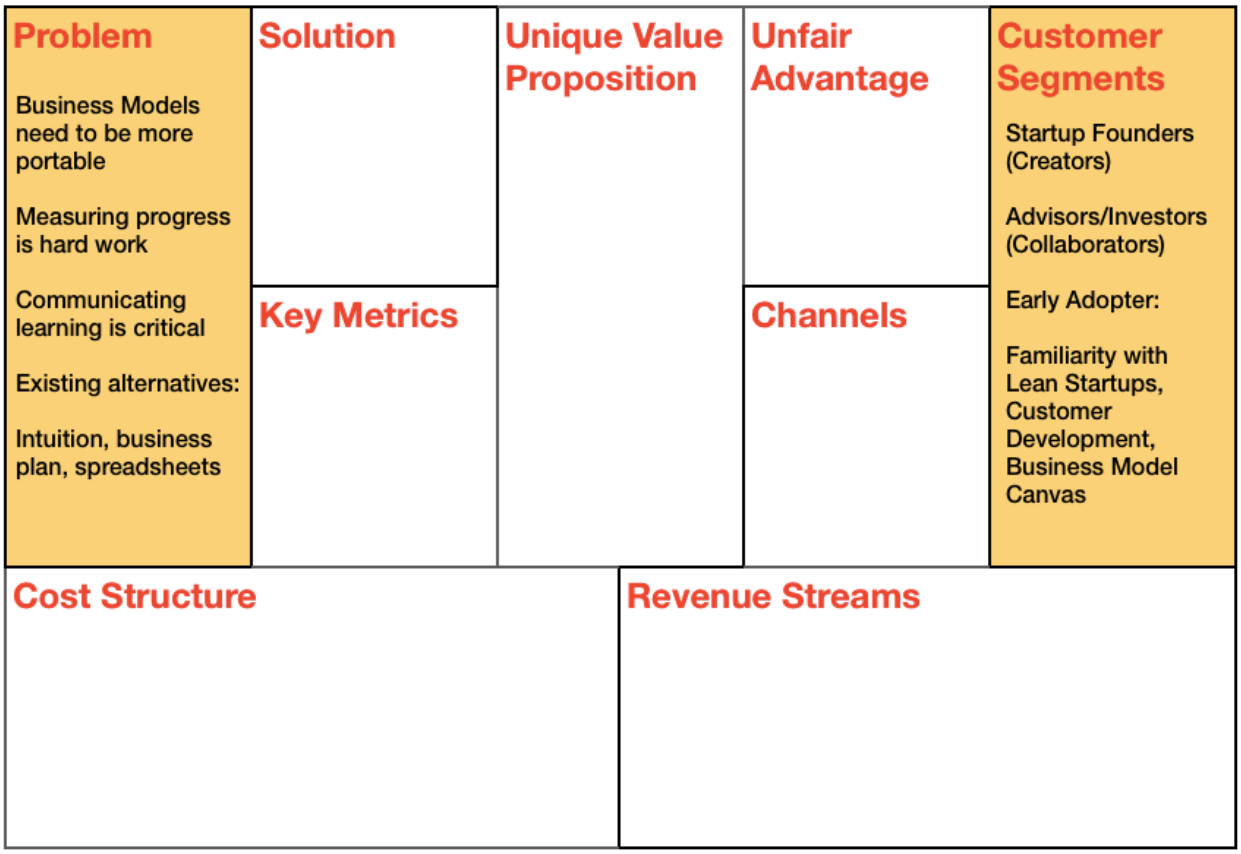
Responsible for documenting hypotheses and communicating progress.

Collaborators (Advisors/Investors) :

Help the founders by providing advice and holding them accountable.

4. Early Adopter

Since the Lean Canvas was a product of synthesizing Customer Development, Lean Startups, and the Business Model Canvas, I believe an early adopter would be someone who has had some familiarity with at least one and ideally all of the above.



Unique Value Proposition

Dead-center in the Lean Canvas is a box for your Unique Value Proposition (UVP). This is one of the most important boxes on the canvas and also the hardest to get right.

"Unique Value Proposition: A single, clear compelling message that states why you are different and worth buying."

- Steve Blank
The Four Steps to the Epiphany

The UVP is hard to get right because you have to distill the essence of your product in a few words that can fit in the headline of your landing page. Additionally, your UVP also needs to be different and that difference needs to matter.

First-time visitors spend 8 seconds on average on a landing page. Your UVP is their first interaction with your product - craft a good UVP and they might stay and view the rest of your site. Otherwise, they'll simply leave.

The good news is you don't have to get this perfect right away. Like everything on the canvas, you start with a best guess and then iterate from there.

How to Craft a Unique Value Proposition

I highly recommend getting a copy of the classic book on marketing by Al Ries and Jack Trout: “Positioning: The Battle For Your Mind”. Ries and Trout are considered the fathers of modern advertising. This is an “easy-read” and the best crash course on marketing I’ve ever come across.

Be different, but make sure your difference matters

The key to unlocking what’s different about your product is deriving your UVP directly from the #1 problem you are solving. If that problem is indeed worth solving, you’re more than halfway there already.

Target early adopters

Too many marketers try to target the “middle” in the hopes of reaching mainstream customers and in the process water down their message. Your product is **not** ready for mainstream customers yet. Your sole job should be finding and targeting early adopters which requires bold, clear, and specific messaging.

Focus on finished story benefits

You’ve probably heard about the importance of highlighting benefits over features. But benefits still require your customers to translate them to their worldview. A good UVP gets inside the head of your customers and focusses on the benefits your customers derive **after** using your product.

So for instance if you are building a résumé-building service:

- a feature might be “*professionally designed templates*”
- the benefit would be an “*eye-catching* résumé *that stands out*”
- but the finished story benefit would be “*landing your dream job*”.

Pick your words carefully and own them

Words are key to any great marketing and branding campaign. Look at how the top luxury car brands have used a single word to define themselves:

Performance: BMW
Design: Audi
Prestige: Mercedes

Picking a few "key" words that you consistently use also drives your SEO ranking.

Answer: What, Who, and Why

A good UVP needs to clearly answer the first 2 questions - what is your product and who is the customer. The "Why" is sometimes hard to fit in the same statement and I'll frequently use a sub-heading for that.

Here are example UVPs I have used in products:

CloudFire - Photo and Video Sharing for Busy Parents.
Get back to the more important things in your life.

USERcycle - Lifecycle Marketing Software.
Turn your users into passionate customers.

Study other good UVPs

The best way to craft a good UVP is to study the UVPs of the brands you admire. Visit their landing pages and deconstruct how and why their messaging works.

Create a high-concept pitch

Another useful exercise is to create a high-concept pitch popularized by VentureHacks in their e-book “Pitching Hacks”. A high-concept pitch usually builds on other familiar concepts to quickly get an idea across and make it easily spreadable. Unlike a UVP, a high-concept pitch is best used in conjunction with something else that sets the right context such as an elevator pitch.

Examples:
- YouTube: “Flickr for video”
- Aliens (movie): “Jaws in space”
- Dogster: “Friendster for dogs”

Case Study: Lean Canvas - Unique Value Proposition

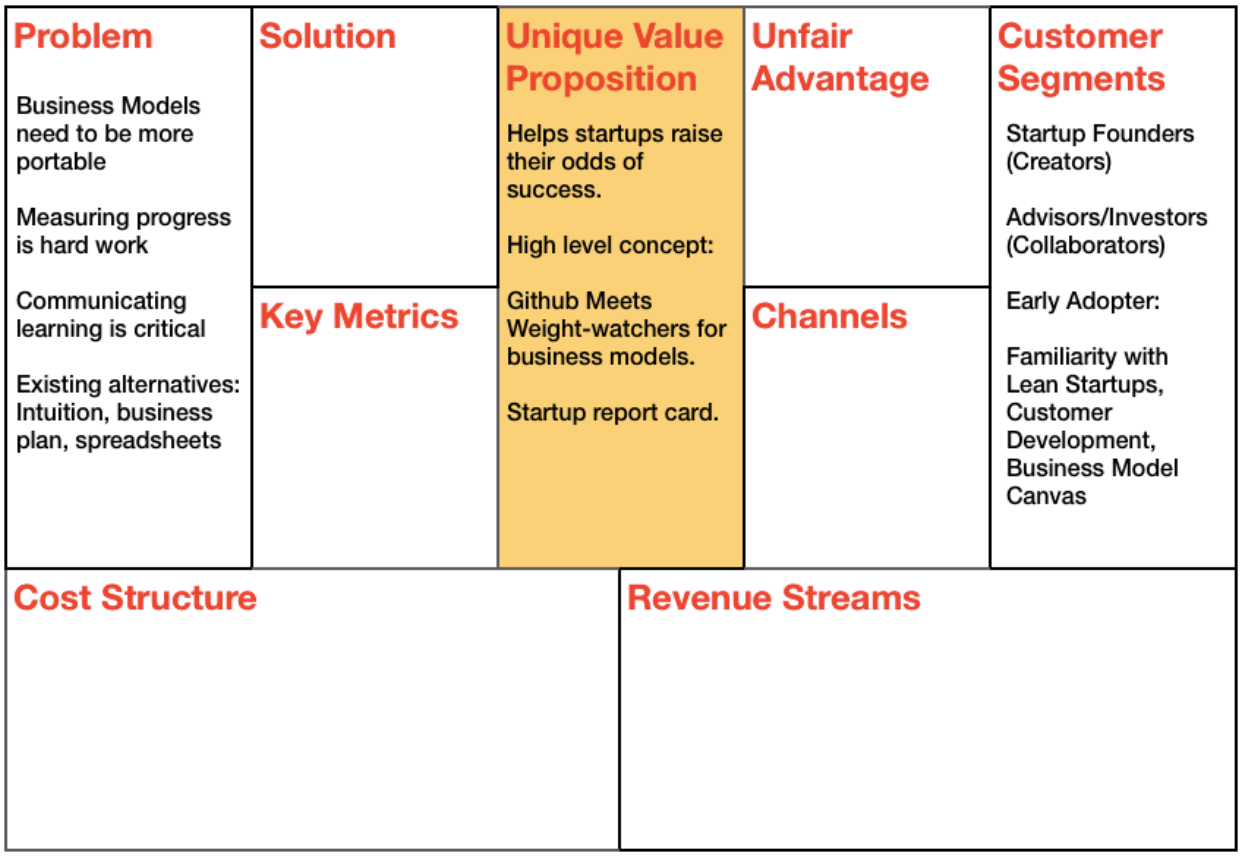
In the case of startups and business models, while Lean Canvas might provide an immediate benefit of offering a better process for testing business models, the finished story benefit is finding a business model that works and eventually building a successful product.

Unique Value Proposition:

Lean Canvas - Business Model Validation Software
Help startups raise their odds for building successful products

I also jot down a few high-level concepts:

- Github Meets Weight-watchers for Business Models
- The Startup Report Card



Solution

You are now ready to tackle solution possibilities.

As all you have are untested hypotheses, I don't recommend getting carried away with fully defining a solution just yet. Rather simply sketch out the top features or capabilities next to each problem.

Bind a solution to your problem as late as possible.

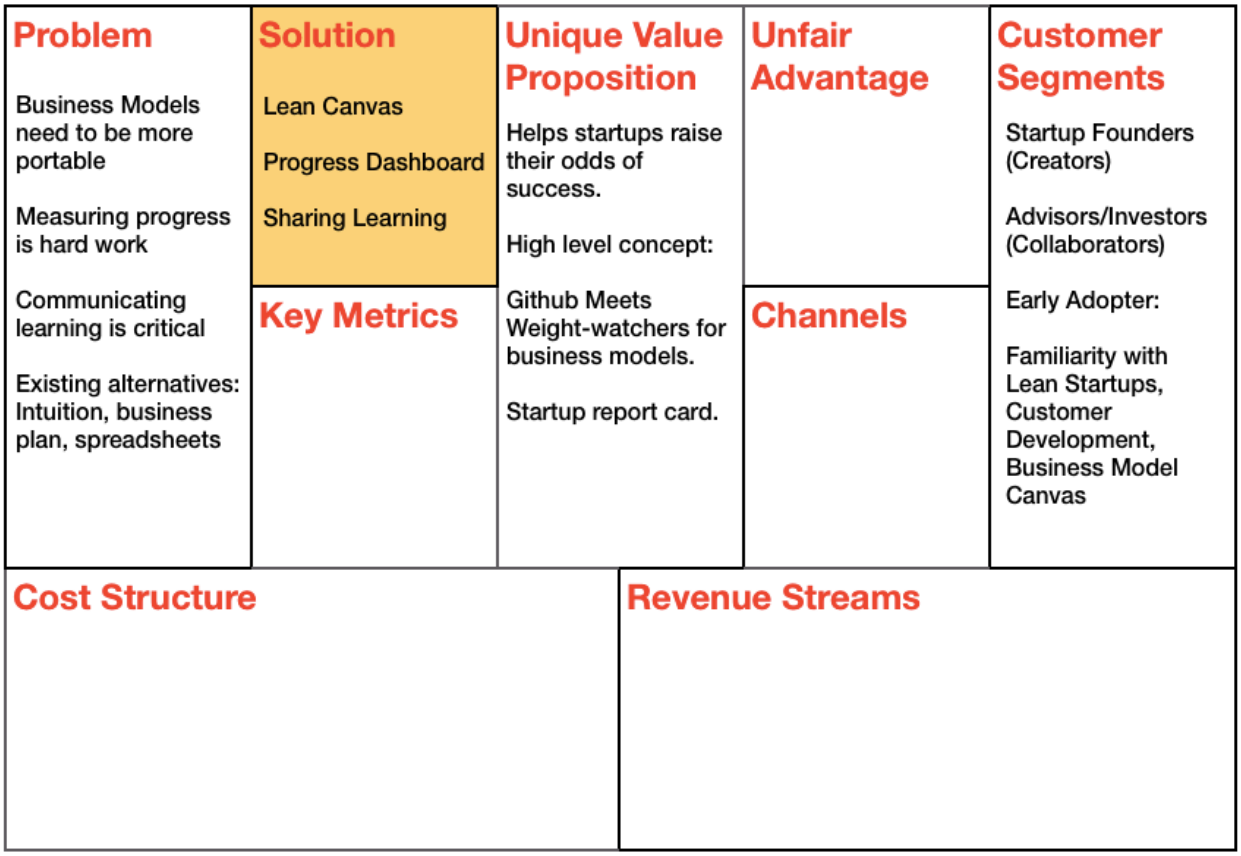
Case Study: Lean Canvas - Solution

For each of the problems outlined earlier, here is a feature or capability that solves the problem:

Problem: Business Models need to be more portable
Solution: Use the Lean Canvas format for capturing hypotheses on 1 page

Problem: Measuring progress is hard work
Solution: Provide a simple way to "dashboard" experiments

Problem: Communicating learning is critical
Solution: Need a sharing feature to facilitate sharing lessons learned



Channels

The good news is that following a "Customer Discovery/Interview" process forces you to build a path to customers early. Unfortunately, unless you are in a direct sales business, that path may not scale beyond Problem/Solution Fit.

In addition to defining the right product to build, it's just as critical to start finding, building and testing a significant path to your customers from day one.

While there are a plethora of channel options available, some channels may be outright inapplicable to your startup, while others may be more viable during later stages of your startup.

I typically look for the following characteristics in my early channels:

FREE'er versus PAID
First of all there is no such thing as a free channel. Channels we normally associate as being free like SEO, social media, and blogging, have a non-zero human capital cost associated with them. Calculating their ROI is complicated because unlike a paid channel which is used up after you pay for it, these channels keep working for you over time.

A commonly cited paid channel is search engine marketing. Eric Ries has written about how he tested his early product on $5 a day using Google Adwords - driving 100 clicks at a CPC of 5 cents. If you can pull this off

today, by all means use it, but unfortunately those days are long gone for most products. Keyword competition is so fierce now that you need to either out-spend or out-wit your competition. Both of these activities are better suited after product/market fit when your focus shifts to optimization versus learning.

Inbound versus Outbound

Inbound channels use "pull messaging" to let customers find you organically while outbound channels rely on "push messaging" for reaching customers.

Example inbound channels: Blogs, SEO, E-books, white papers, webinars.
Example outbound channels: SEM, print/TV ads, trade shows, cold calling.

When you don't yet have a tested value proposition, it's hard to justify spending marketing dollars or effort on outbound messaging. Getting "techcrunched" or seeking other forms of PR before then is a form of waste. Now might be the time to start building inroads to influencers but you are not ready to "get covered".

Direct versus Automated

As a scalable channel, direct sales only make sense in businesses where the aggregate lifetime value of the customers exceeds the total compensation of your direct salespeople - such as in certain B2B and Enterprise products.

But as a learning channel, direct sales is one of the most effective since you interact face-to-face with the customer. The Customer Discovery process we'll cover in a little bit is direct selling repurposed for accelerating learning from customers.

First sell manually, then automate.

Direct versus Indirect

Another area where startups waste energy is prematurely trying to establish strategic partnerships. The idea is to partner with a larger company to leverage their channels and credibility. The problem is that until you have a proven product, you won't get the right level of attention from the bigger company's sales reps to make this work. Given the choice of selling what you know or an unproven product to make your quota, which would you choose?

First sell yourself, then let others do it.

The same principle allow applies to hiring external salespeople. While a salesperson can probably outsell you on the execution of a sales plan, they can't create that plan.

Retention before Referral

Many startups are obsessed with building virality and referral/affiliate programs into their product from day 1. While referral programs can be very effective in spreading the word about your product, **you need to have a product worth spreading first.**

"Build a remark-able product."

- Seth Godin
Purple Cow

Building the ideal early channel

An early channel I recommend building that delivers on all the points above is **Content Marketing**.

Content Marketing uses a combination of Content, Search Engine Optimization (SEO), and Social Media to work. Rather than crafting the "perfect outbound message", you instead incrementally test various aspects of your Problem/Solution using inbound channels like blogs, white papers, and webinars. SEO and Social Media serve to further enhance the reach of your content.

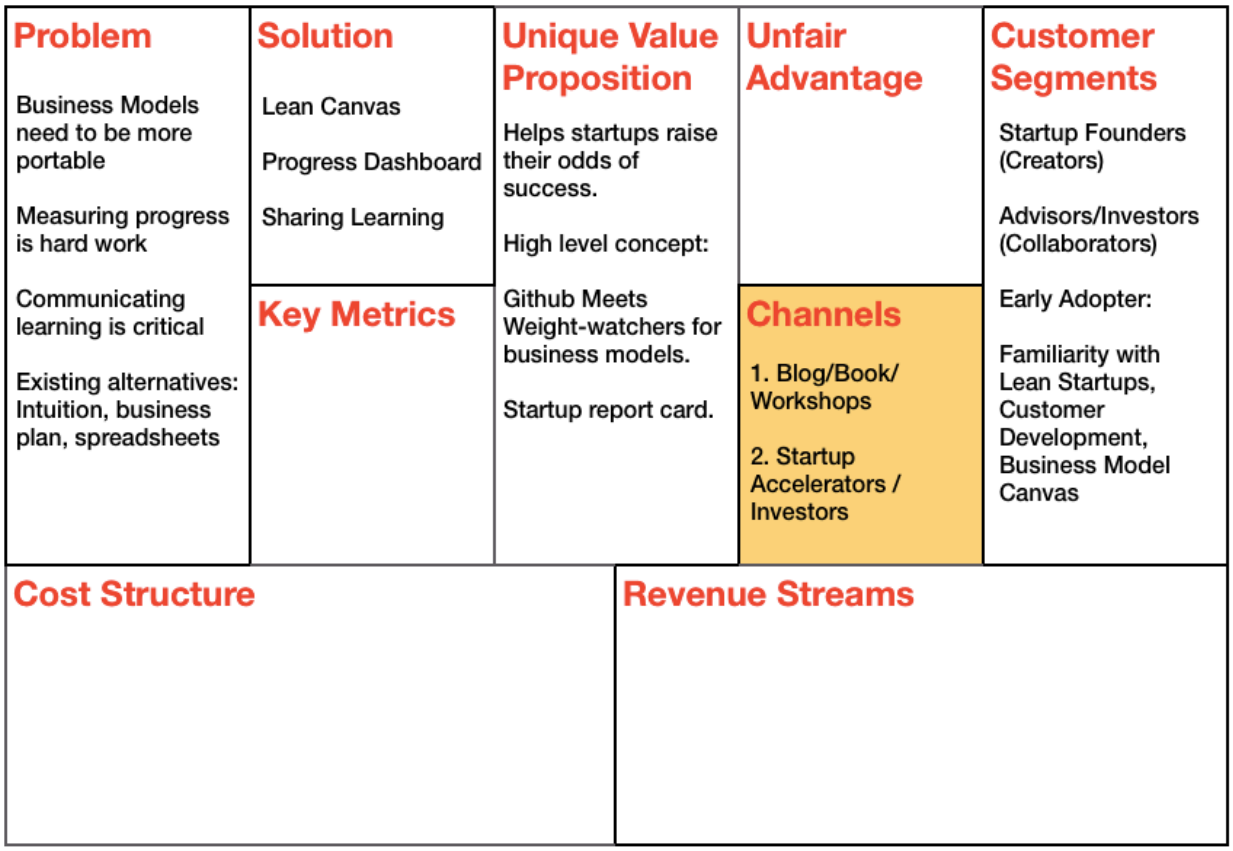
Content Marketing isn't free, takes time to build, and does cost time. But when it starts to work, "Content Marketing" turns from an expense into an asset. It could even become your "Unfair Advantage".

In the next chapter, I'll cover more specific details on how to get started with "Content Marketing".

Case Study: Lean Canvas - Channels

While the problems I outlined above were based on my own experiences, I witnessed other startup founders recount similar stories in the workshops I was teaching. That prompted me to write about Lean Canvas on my blog and also test it in a presentation I gave at Capital Factory's Demo Day in Austin to a mostly investor audience. Both were well received which pushed me towards incorporating Lean Canvas in my workshops. Lean Canvas was tested and refined in these presentations and workshops before it ever became an online tool.

My blog, workshops, and book are natural early channels to reach other startup founders interested in Lean Startup/Customer Development.



Revenue Streams and Cost Structure

A lot of startups choose to defer the “pricing question” because they don’t think their product is ready. Something I hear a lot is that a minimum viable product is by definition embarrassingly minimal. How can you possibly charge for it?

First, a minimum viable product is not synonymous with a half-baked or buggy product. Your MVP should address the top problems customers have identified as important to them. By that definition, it should deliver enough value to justify charging.

But, there is another line of reasoning frequently cited for deferring pricing - to accelerate initial learning. The argument goes that pricing creates unnecessary friction that should be avoided early on.

While this strategy may work (especially if you are already funded), I find that it delays testing one of the riskier and critical parts of your business model. Furthermore, I have found that you don’t need a lot of users to support learning. You need just a few good customers.

I believe that if you intend to charge for your product, it’s better to be upfront about it. It sets the right expectations, raises customer commitment, starts generating cash flow, and lets you start tackling one of the riskier parts of your business model early.

What you charge for your product is simultaneously one of the most complicated and most important things to get right. Not only does your pricing model keep you in business, it also signals your branding and positioning. It determines your customers.

Your price is part of your product.

Although there is a lot of science around pricing, pricing is more art than science. For a great primer, I highly recommend getting a copy of Neil Davidson's free e-book on software pricing: "Don't Just Roll the Dice".

Pricing is not unlike any other business model hypothesis and should be tested using the same criteria we covered earlier when building validated learning loops - create a testable hypothesis, time-box the experiment, and validate qualitatively, then verify quantitatively.

The strategy I've found that meets all these criteria is **starting with a single "Free Trial" pricing plan**.

Let's see why.

Start with a single pricing plan

Starting with multiple plans that cover everyone under the sun is a form of waste. I've seen startups launch with plan options targeting 1 person startups to 1000+ person enterprises.

Not only does supporting multiple plans require you to write more code to support plan/feature segmentation, but the return on learning is diluted when you attempt to target multiple customer segments all at once. In the example above, the business models and tactics vary greatly when selling to startups than to enterprises.

The bigger point here though is that when you're starting out, you don't yet have enough information to know how to correctly price or segment the feature set into multiple plans.

Use a "Free Trial" plan

Time-based trials help time-box your pricing experiments so you can force a conversion decision which allows you to learn and iterate faster.

Pick a price to test

Existing alternatives create "reference points" in the minds of customers that they will use to rank your solution so it's important to understand and position your price against them.

In the rare case you are actually solving a brand new problem or don't have clear reference points (more common in Enterprise based products), you might have to pick a starting price out of thin air and refine from there.

"Pricing is all about setting the right perception."

- Neil Davidson
Don't Just Roll the Dice

Take your costs into account

The ultimate goal is finding a scalable business model so it should go without saying that you also need to keep an eye on what it would cost you to deliver your solution and ensure you have a healthy margin built in.

One rule of thumb for building a successful business is ensuring the lifetime value of your customers exceeds the cost of customer acquisition by at least a factor of three.

It's hard to accurately calculate these at this stage so instead do a back-of-the-envelope calculation based on your people/hardware costs and subscription revenue to find your break-even point.

What about Freemium?

Freemium is a popular model used by numerous web applications. It was first popularized by Fred Wilson on his blog where he described it as:

> *"Give your service away for free, possibly ad supported but maybe not, acquire a lot of customers very efficiently through word of mouth, referral networks, organic search marketing, etc., then offer premium priced value added services or an enhanced version of your service to your customer base."*
>
> - Fred Wilson
> AVC Blog

On the surface, Freemium seems like the best of both worlds: Get users to try your service without having to worry about price, then up-sell them into the right premium plan later. But the reality is quite different.

First, I believe that unless you are deriving monetary value from free users, the Freemium model is less of a business model and more of a marketing tactic to fill your pipeline with potential prospects.

Second, I believe pricing is one of the riskiest (and most critical) parts of the business model and should be tested early. Freemium delays this learning.

While I agree Freemium can be a highly effective model, I don't advocate starting with it.

The Problem with Freemium

Low or no conversions

Many services make the mistake of giving away too much under their free plans, which leads to very low or no conversions. One reason for this is that creatives (artists, musicians, developers) are especially known to undervalue their own work and are really bad at setting price.

Pricing should be set with the buyer in mind, not the seller.

But the main reason is something we covered in the last section. You don't yet have enough usage data to correctly define the FREE plan so that users naturally outgrow it at some predictable time in the future.

Long validation cycle

Even the best Freemium services report conversion rates in the 0.5-5.0% range which leads to long validation cycles. Time is the most valuable resource for a startup and you can't afford such long learning cycles on something as critical as price.

Focus shifts to the wrong metric

Because "FREE" can be irrationally appealing, Freemium has a tendency to cause a premature shift in focus from user retention to user acquisition (sign-ups). Unless you have built the right product, getting more sign-ups is waste. You don't need a lot of traffic to build the right product - just the right initial customers.

Your free users are not your customers (yet).

Low signal-to-noise ratio

When you have a lot of free users, it's hard to focus your attention on the right feedback.

Given the opportunity, everyone can be a critic.

Free Users aren't "free"

Even though the operational costs of carrying a free user may seem low, they aren't zero. Other than server bandwidth/hosting costs, there are support, feature, and learning costs (like the ones described above) that need to be taken into account.

Lincoln Murphy described a quid pro quo test in his paper: "The Reality of Freemium in SaaS" for valuing free users. Unless free users are adding participatory value (as found in services with high network effects like Linked In, Facebook, and twitter), they are an expense.

Jason Cohen, who writes the popular "A Smart Bear" blog, even advocates accounting for free users as a "marketing expense" on your balance sheet much like you would an ad-buy, or trade show expense.

How to Approach Freemium

Start with the premium part of Freemium first

Once you recognize Freemium as a marketing tactic and make a conscious decision to shorten the validation cycle, it makes sense to start with the premium part of Freemium first and use a single pricing plan your customers will bear.

Since your eventual goal is to charge for your product anyway, why not start there? Pick features and a plan based on what customers will pay for today and sign them on as your first customers. Not only is this simpler to build but it's also simpler to measure.

Then, once you have learned how your customers are using your product, you can always offer a Free plan if you want to. You would have collected valuable usage data along the way which puts you in the best position to design multiple upstream and downstream plans.

Case Study: Mailchimp

Mailchimp is frequently cited as one of the Freemium model success stories, but too often people fail to recognize that Mailchimp didn't start with a free plan. In fact they spent years building a powerful, affordable (but not free), profitable product first, with years of pricing experimentation, before backing into a free plan.

What is a good Free plan?

A good Free plan should ideally behave similarly to a Free Trial. The difference is that while a Free Trial is time-based, Freemium is usage-based. If you understand the usage pattern of your product, you should be able to design the Free plan so that a user naturally outgrows it at some point X in the future that you can reasonably predict.

At that point the difference between Freemium and Free Trial is the perception of offering something FREE which is a big enough difference to warrant the use of Freemium for certain types of products.

When to use Freemium versus Free Trials?
Once you've built the right product, Freemium can be a powerful user acquisition strategy for consumer facing products that naturally tend to be more "FREE" driven.

Businesses, on the other hand have come to expect time-based trials and the added complexity of tracking and carrying free users may not be warranted here.

Case Study: Lean Canvas - Revenue Streams and Cost Structure

Revenue Streams

Lean Canvas pricing will be largely determined by who will ultimately be the customer - bootstrapped founders, funded founders, larger companies, investors, etc.

For the case of startup founders, they tend to be quite price sensitive and given the fact that their existing alternatives are all free, I decide to start with the following:

"A 30-day Free Trial plan priced at $14/mo that lets them create 1 canvas and invite up to 3 collaborators."

Cost Structure

In my blog post on Lean Canvas, I made a public call for help building out this tool and got a dozen responses from developers around the world. I selected 3 people

from this list and am currently building Lean Canvas with them - Lukas Fittl (Austria), Ross Hale (Santa Barbara), and Andrew Elliott (Santa Barbara). They are all seasoned entrepreneurs in their own right and have a passion for Lean Startups.

We came to an agreement to spend just enough effort to get the Minimum Viable Product built over 2 weeks and evaluate how we move forward after that. We agreed not to pay ourselves till the service had positive cash flow.

That said, I still used the following analysis to figure out the opportunity cost for building and testing the MVP and more importantly determine my break-even point.

Costs

Problem/Solution Fit:

2 interviewers * (50 interviews * 30 min per interview) = 50 hours

Effort to get ready for interviews (mockups, scripts, etc.) = 40 hours

Build MVP:

2 developers * 2 weeks * 20 hours per week = 80 hours

Cost to launch = 170 hours * $65/hr = $11K

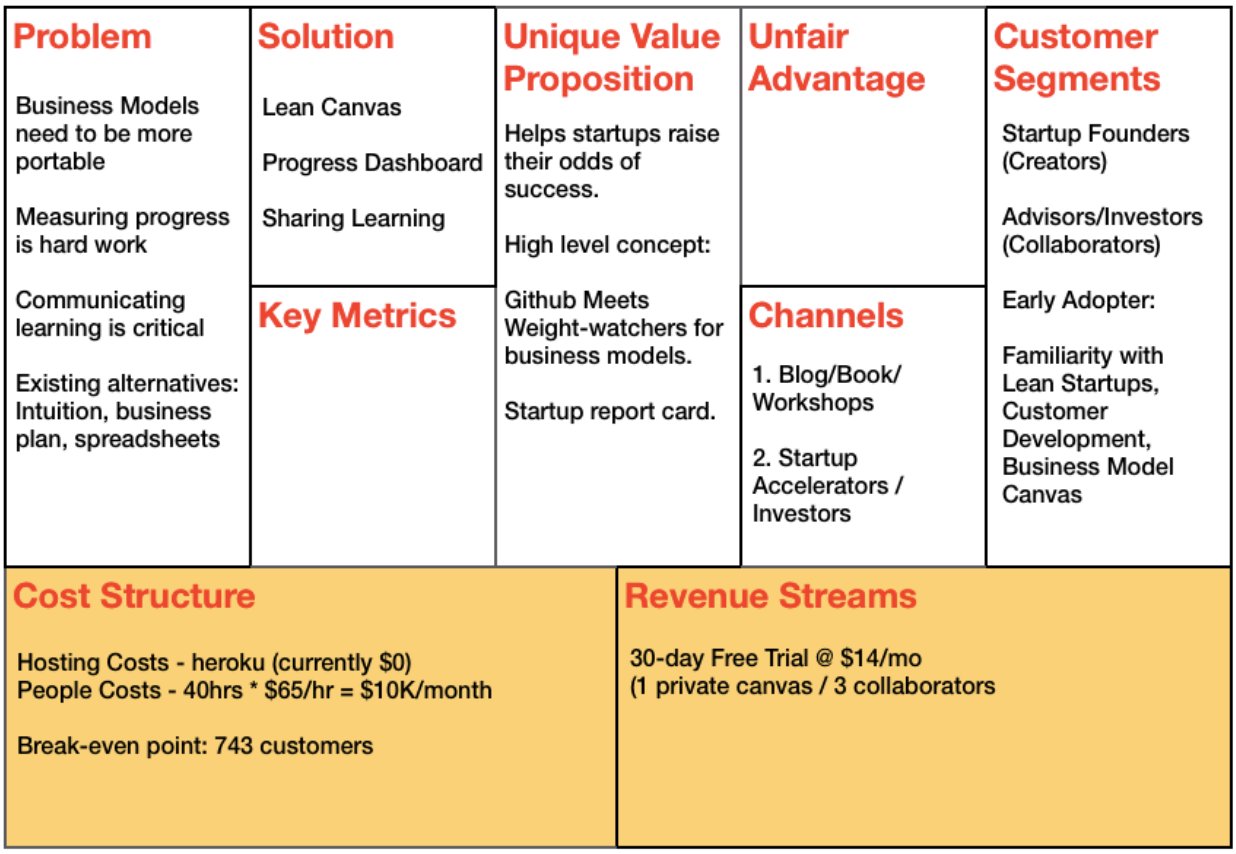
Ongoing People costs = 4 ppl * 10 hours per week * $65/hr = $10,400/mo

Hosting costs currently $0 (thank you heroku)

Revenue

Break-even point @$14/mo = $10,400K / $14/mo = 743 paying customers

Break-even point @$24/mo = $10,400K / $24/mo = 434 paying customers



Key Metrics

"Find the key number that tells you how your business is doing in real time, before you get the sales report."

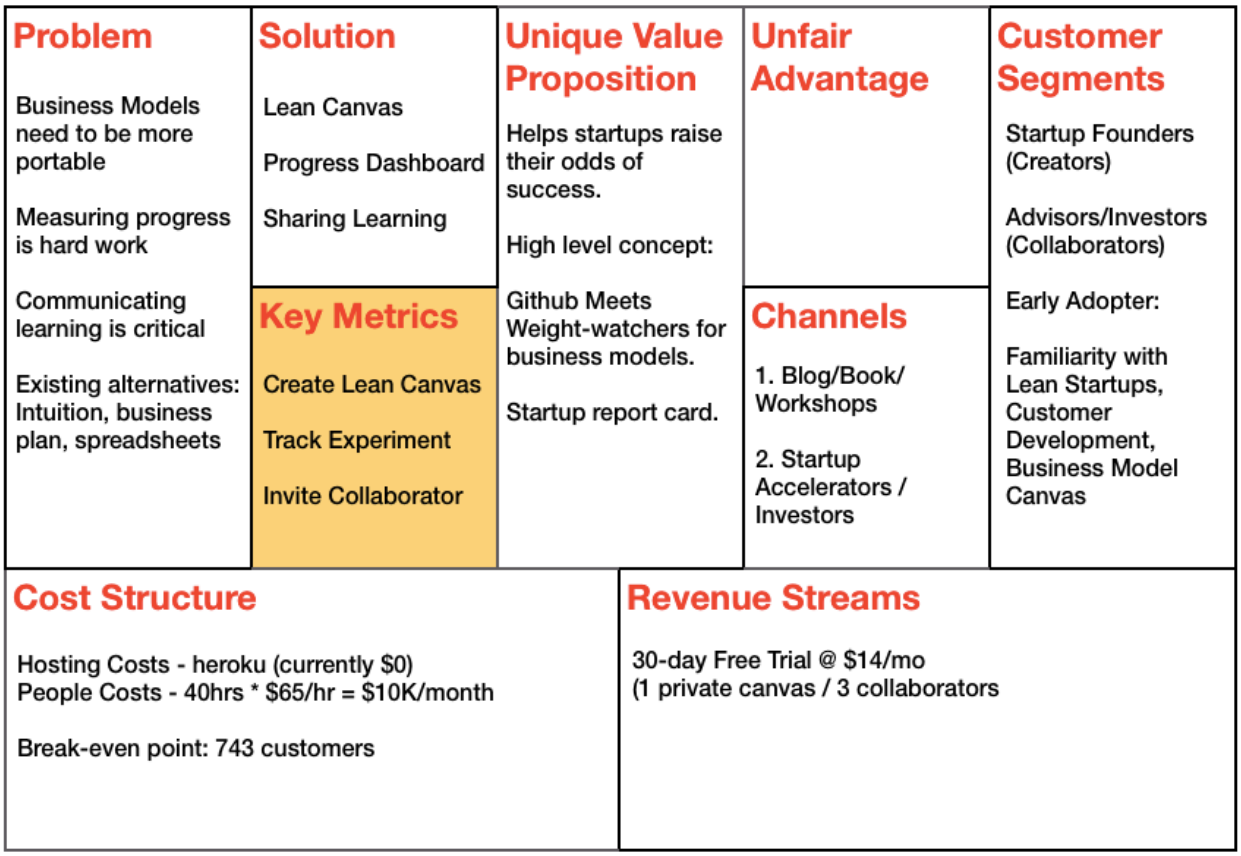
- Norm Brodsky and Bo Burlingham
The Knack

Startups are inherently chaotic, but fortunately, there are only a handful of key metrics that drive a web startup. We'll cover five key startup metrics (Dave McClure's Pirate Metrics) a little later but for right now document the one or two key activities you think will drive usage of your product.

So for example, if you are a blogging platform writing a blog post would be the key activity.

Case Study: Lean Canvas

Creating a lean canvas is the first key activity users would complete but using the tool to **track running experiments** is the ongoing key activity that will drive use of the tool.



Unfair Advantage

This is usually the hardest section to fill which is why I leave it for last. Most founders list things as competitive advantages that really aren't. An interesting perspective, via Jason Cohen, to keep in mind is that anything worth copying will be copied. So imagine your co-founder steals your source code, sets up shop in Costa Rica, and slashes prices. Do you still have a business?

You have to be able to build a successful business in spite of that which leads to the following definition:

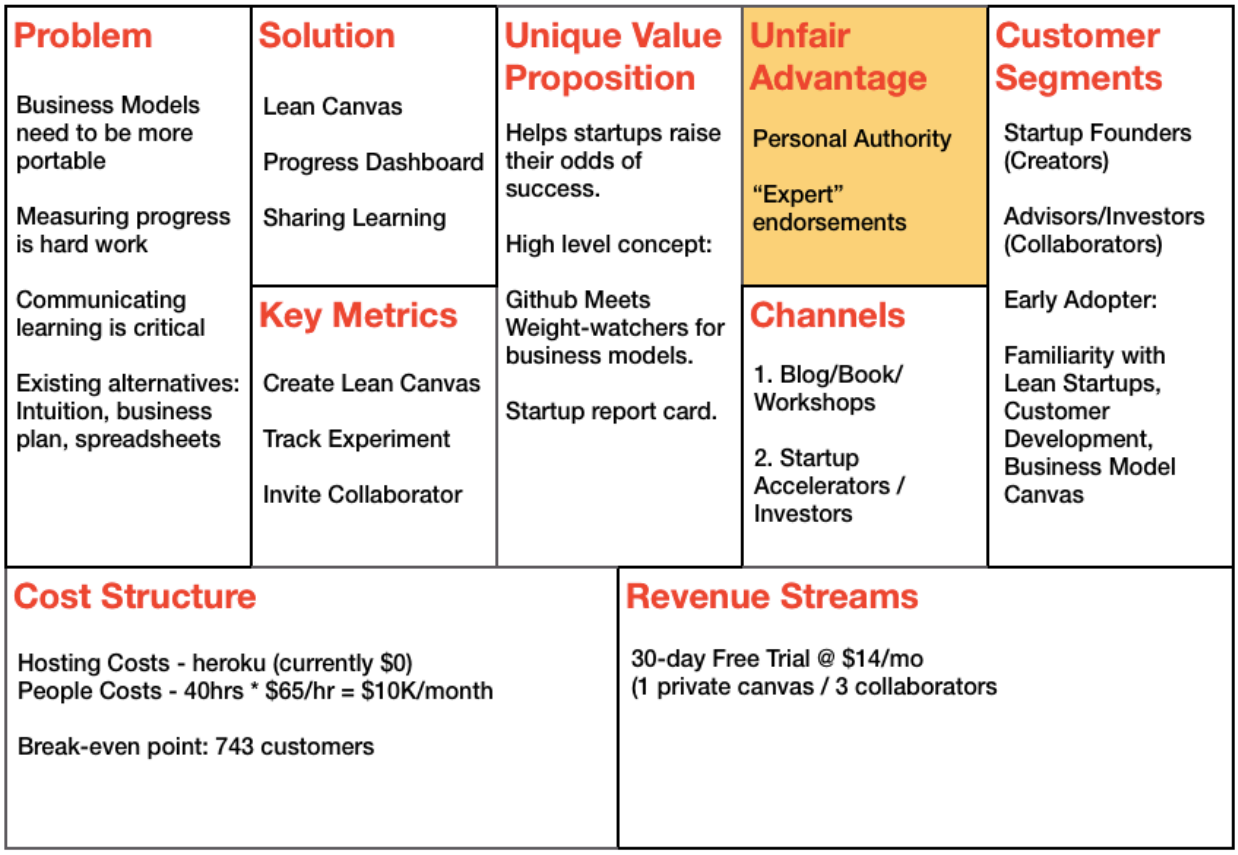
"A real unfair advantage is something that cannot be easily copied or bought."

- Jason Cohen
A Smart Bear

You may initially have to leave this box blank but it's here to have you really think about how you can make yourself different and make your difference matter. Some examples of unfair advantages - insider information, the right "expert" endorsements, personal authority.

Case Study: Lean Canvas

In the course of writing my blog and book, I have been able to attract a growing and active audience of readers and "experts" interested in Lean Startups which has taken me over a year to build. Only time will tell if this "reputation" and channel proves to be an unfair advantage.



Problem

Business Models need to be more portable

Measuring progress is hard work

Communicating learning is critical

Existing alternatives: Intuition, business plan, spreadsheets

Solution

Lean Canvas

Progress Dashboard

Sharing Learning

Key Metrics

Create Lean Canvas

Track Experiment

Invite Collaborator

Unique Value Proposition

Helps startups raise their odds of success.

High level concept:

Github Meets Weight-watchers for business models.

Startup report card.

Unfair Advantage

Personal Authority

"Expert" endorsements

Channels

1. Blog/Book/ Workshops

2. Startup Accelerators / Investors

Customer Segments

Startup Founders (Creators)

Advisors/Investors (Collaborators)

Early Adopter:

Familiarity with Lean Startups, Customer Development, Business Model Canvas

Cost Structure

Hosting Costs - heroku (currently $0)
People Costs - 40hrs * $65/hr = $10K/month

Break-even point: 743 customers

Revenue Streams

30-day Free Trial @ $14/mo
(1 private canvas / 3 collaborators

Prioritize Where to Start

Once you have your Lean Canvases sketched for each customer segment, lay them side by side and select the best business model to start with.

Your objective is to find a **big enough market** you can **reach** with customers who **need your product** that **will pay a price** you can build a business around.

Here is the weighting order I use (ranked from highest to lowest):

1. Customer pain level
Pick customer segments that need your product the most. The goal is to have one or more of your top three problems as must-haves for them.

2. Ease of reach
Building a path to customers is one of the harder aspects of building a successful product. If you have an easier path to one segment of customers over others, take that into consideration. It doesn't guarantee you'll find a problem worth solving or a viable business model, but it will get you out of building faster and speed up your learning.

3. Price
What you can charge for your product is largely driven by the customer segment. Pick a customer segment that allows you to maximize on price. The more you can charge, the fewer customers you need to reach break-even (assuming your gross margins don't change).

4. Market Size

Lastly, you need to pick customers that represent a big enough market, or are a stepping stone to a big enough market, that you can build a business around.

Case Study: Lean Canvas

I created other canvases for my different segments (not shown here) and while I could potentially derive higher pricing with other customer segments, I decided to start testing the "Startup Founder" segment because it was the one I best understood and easily reach.

Now it's Your Turn

Documenting your Plan A is a prerequisite for moving on. Too many founders carry their hypotheses in their heads alone which makes it hard to systematically build and test a business.

You have to draw a line in the sand.
How you create your Lean Canvas is up to you.

You can

- Visit http://LeanCanvas.com and create your online canvas there
- Create a version in Powerpoint or Keynote
- Sketch a canvas on paper

The important thing is sharing it with at least one other person when you are done.

Note: The boxes on the Lean Canvas are intentionally small because it forces you to be concise. You may find it easier to free-form your answers first like I did in the last section and then fill out the canvas.

CHAPTER 3

GET READY TO INTERVIEW CUSTOMERS

The fastest way to learn is by talking to customers. Decouple the problem from the solution and test the problem before binding yourself to a solution.

No Surveys or Focus Groups Please

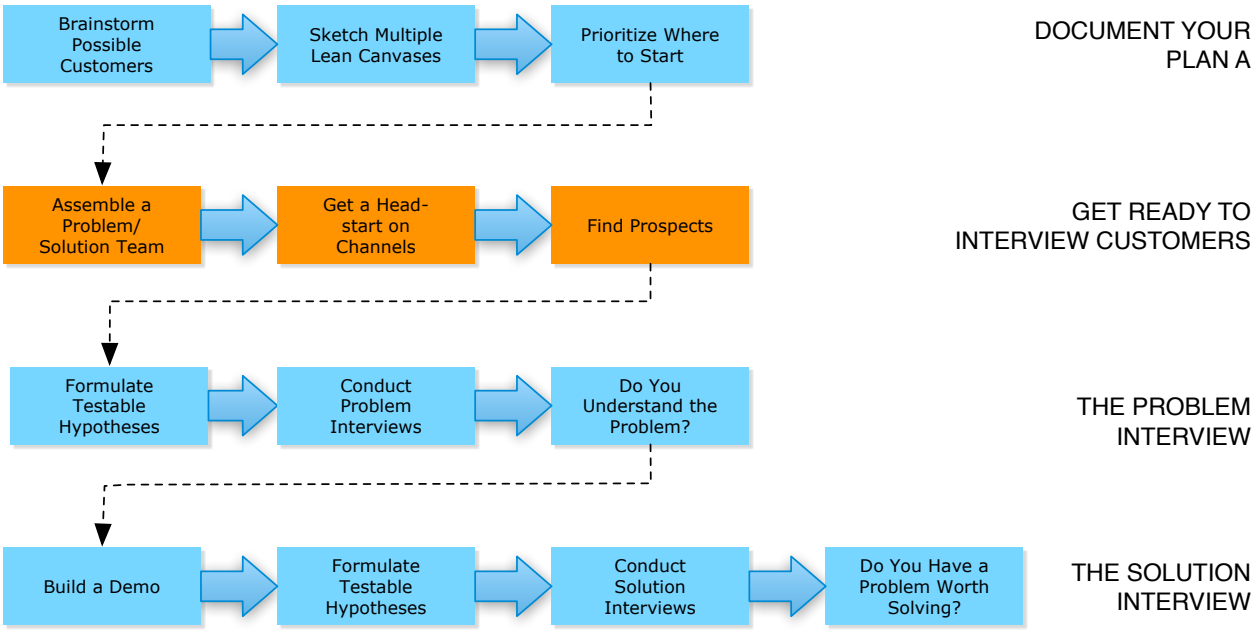
Get Out of the Building

Assemble a Problem/Solution Team

Get a Head-start on Channel Building

Find Prospects

Decouple the Problem from the Solution


Brainstorm Possible Customers
Sketch Multiple Lean Canvases
Prioritize Where to Start
DOCUMENT YOUR PLAN A
Assemble a Problem/ Solution Team
Get a Head-start on Channels
Find Prospects
GET READY TO INTERVIEW CUSTOMERS
Formulate Testable Hypotheses
Conduct Problem Interviews
Do You Understand the Problem?
THE PROBLEM INTERVIEW
Build a Demo
Formulate Testable Hypotheses
Conduct Solution Interviews
Do You Have a Problem Worth Solving?
THE SOLUTION INTERVIEW

No Surveys or Focus Groups Please

When asked to do the smallest thing to learn from customers, the first instinct of many founders is to conduct a bunch of surveys or focus groups. While running surveys and focus groups may seem more efficient than interviewing customers, starting there is usually a bad idea.

Here's why:

Surveys assume you know the right questions to ask
It is hard, if not impossible, to script a survey that hits all the right questions to ask because you don't yet know what they are. During a customer interview, you can ask for clarification and explore areas outside your initial understanding.

Customer Discovery is about exploring what you don't know you don't know.

Worse, surveys assume you know the right answers too
In a survey, you not only have to ask the right questions but also provide the customer with the right choice of answers. How many times have you answered a survey where your best answer was "Other"?

The best initial learning comes from "open-ended" questions.

You can't see the customer during a survey
Body language cues are as much an indicator of Problem/Solution Fit than the answers themselves.

Focus Groups are just plain wrong
The problem with focus groups is that they quickly devolve to “group think” which is wrong for most products.

Are Surveys Good For Anything?

While surveys are bad at supporting initial learning, they can be quite effective in verifying what you learn from customer interviews.

I discussed the principle of two-phase validations earlier - **first qualitative, then quantitative**. The customer interview is a form of qualitative validation that is quite effective in uncovering strong signals for or against hypotheses using a “reasonably” small sample size.

Once you have preliminary validation on your hypotheses, you can then use what you have learned to craft a survey and verify your findings quantitatively. The goal is no longer learning but demonstrating scalability (or statistical significance) of the results.

Get Out of the Building

The smallest thing you can do to learn from customers is talk to them. Not releasing code, or collecting analytics, but talking to people.

Release Early, Release Often in not enough

"Release early, release often" was a mantra software developers jumped on several years ago as a means to facilitate faster feedback, but spending any time building even this "small" release can be time wasted. First, these "small" releases are almost never "small" enough. But more importantly, it's hard building something people want without involving them in the process. Yes, you sometimes have to "show something" for others to get it but you don't need code for that - proxies like mockups, videos, and landing pages fit the bill quite well.

Case Study: Dropbox

While building Dropbox at MIT, Drew Houston posted a short 3 minute demo on Hacker News which went viral. The video plus teaser landing page helped him attract tens of thousands of early adopters, find a co-founder, and get accepted into Y Combinator. At the time Drew estimated launch being less than 3 months away. It took him 18 months to publicly launch Dropbox.

Metrics can't explain themselves

Another commonly used tactic is to put up a teaser page, drive some traffic to it, and measure the results. Metrics can only tell you what actions your visitors are taking (or not), but not why. Did they abandon because of your copy, graphics, pricing, something else? You could endlessly try various combinations, or you could just ask the customers.

But talking to people is hard

There are email people and there are phone people.

I used to be a “closeted technical founder”. With my first product, I preferred that customers send me their feedback over email rather than over the phone.

After listening and acting on too many wrong requests, I realized that just seeking customer feedback isn’t enough - you have to know how to process this feedback. I have done a complete reversal to where now my mobile phone is wired to the 1-800 number of my products (more on this later).

Customer interviews, usability tests, feature prioritization, etc. are all tactics we’ll cover in subsequent sections that provide a way for getting actionable feedback from customers.

Hack: Scratch Your Own Itch

(or why I don't need to interview customers)

The 37signals folks advocate building software for yourself (i.e. being your own customer), as the best way to build a successful product. While I agree that is an advantage since you start with a problem you've experienced first-hand, it's **not an excuse** for not talking to customers. For starters, can you really be that objective about the problem and pricing with yourself?

Unless you are building products for other entrepreneurs that share the same worldview as yourself, **you are not your customer**. Even if you think you are building products for other entrepreneurs that share the same worldview as yourself, **you have to test that**.

Case Study: CloudFire

I started building my last product, CloudFire, for myself. The itch I was scratching was that I wanted to be able to access all my media (photos, videos, music) from any device and selectively share some if it with others. Sharing (uploading) lots of large media files is a hassle and our solution allowed you to do this rather painlessly (without any uploading).

However, in the process of applying Customer Development I found a much more addressable target market that clearly wasn't me: busy moms and wedding photographers.

Scratching your own itch is a great way to get started but you still need to validate that you have a problem worth solving by talking to other people.

Assemble a Problem/Solution Team

Before you start running your first set of experiments, it's important to assemble the right team so you maximize for speed and learning.

Forget Traditional Departments and Titles

In a Lean Startup, traditional department labels like Engineering, QA, Marketing, etc. can get in the way and create needless friction. Eric Ries instead recommends organizing around 2 teams - the problem team and the solution team.

The Problem Team

The Problem Team is *mostly* involved with "outside the building" activities such as interviewing customers, running usability tests, etc.

The Solution Team

The Solution Team is *mostly* involved with "inside the building" activities such as writing code, running tests, deploying releases, etc.

I say *mostly* because these teams need to be highly cross-functional with overlapping members. Also, interacting with customers is everyone's responsibility.

While I agree with the logical distinction between problem and solution teams, at this stage of a product you're best served with having a single Problem/Solution team.

Start With the Smallest Team Possible, But No Smaller

The ideal problem/solution team size is 2-3 people.

There are many arguments for building your Release 1.0 (Minimum Viable Product) with a small team:

- Communication is easier
- You build less
- You keep costs low

I built my last product, CloudFire, "mostly" as a single founder. The biggest challenge I faced was balancing outside the building activities with inside the building activities and I had to come up with a set of work hacks to make this work (See Appendix: How to Achieve Flow in a Lean Startup).

While it is possible to build a product by yourself, I highly recommend surrounding yourself with at least one other person that can at a minimum help enforce periodic reality checks. Ideally this is a co-founder but advisors, investors, and even an ad hoc board made up of other startup founders can also help fill this role.

The more important thing is not the number of members but ensuring you have the right talents within the team to iterate quickly.

The 3 Must-haves: Development, Design, Marketing

You don't always need 3 people to complete the team. Sometimes you can find these talents across 2 people, sometimes all you need is 1. I tend to look for people with some level of all three.

Here's how I define them:

Development
It goes without saying that if you are building a web application you need strong development skills on your team. Having prior experience building stuff is key along with strong expertise in the specific technology/language you are using.

Design
By design, I mean both aesthetics and usability. In newer markets, function can take precedence over form but we live in an increasingly "design-aware" world where form cannot be ignored. Also, a product is not just a collection of features but rather a collection of user flows. You need people on your team that can deliver on the right experience that matches your customers' worldview.

Marketing
Everything else is marketing. Marketing drives the external perception of your product and you need people that can put themselves in the shoes of your customer. Good copywriting and communication skills are key here along with an understanding of metrics, pricing, and positioning.

Be Wary of Outsourcing your Problem/Solution Team

I constantly run across teams that try and outsource one or more of these 3 areas which is usually a bad idea. When you outsource even one of development, design, or marketing, you are at the mercy of someone else's schedule which can limit both your ability to iterate quickly and learn.

Get a Head-start on Channel Building

A benefit of retrospective learning is that you can go back and reorder steps for efficiency which is what I've done here. Building and testing a path to customers is not only a slow process, but it is imperative that you start doing it as early as possible.

In order to build a successful product, you have to eventually find a scalable and repeatable way to reach customers. There is an implicit expectation that customer development will uncover that path to customers. My experience (with web based products) has been that it's not as much the uncovering of the path but the **building of the path** that is troublesome. Some paths are obvious but hard, such as those built on referrals (word of mouth), SEO, etc.

It's comparatively a lot easier to find 30-50 people to interview, validate you have a problem worth solving, build a MVP, even get them to pay you – **all of which is a false positive** if it was predicated on a customer acquisition approach that won't scale or more importantly be applicable to how you acquire customers in the future.

The primary distribution channel for most web applications is through a website and you need to start laying the groundwork now to building this channel and establishing some baseline traffic you can use later.

Create a Landing Page

With a UVP in hand, you are now ready to put up a basic teaser landing page. The main purpose of this landing page is to start testing your UVP and build a list of potential prospects you could interview.

Establishing a website early with the right "key" words from your UVP will also give you a head start building your SEO ranking. Don't worry about giving away too much about your product. We're only going to mention the "Problem" - not the "Solution".

The key here is starting simple. We'll have ample time to refine this teaser page into a full-fledged website later.

Here's how to get started:

Pick a product name

This is probably the hardest part of this exercise mostly because it is so difficult to find .com domain names that aren't already taken. That said, if you do stumble upon the right "key" words while brainstorming your UVP you might get lucky like I did:

> **Lean Canvas**: Business Model Canvas + Lean Startup
> **USERcycle**: User Lifecycle Marketing Software

Make sure the twitter handle and Facebook page are available

If you are able to register the .com domain, you will more than likely be in the clear with everything else. Register them now even if you don't intend on using them right away.

Keep it simple at first - Just State your UVP

Your UVP will be one of the most important elements of your finished landing page and it's all you need to put on a teaser page. The objective right now is grabbing attention by articulating a problem that resonates with your visitors - not pitching your product.

Follow basic SEO practices

Make sure you also use your UVP in your title tag and place your "key" words (not your product name) early.

For example,

Use this: Lifecycle Marketing Software - USERcycle

Not this: USERcycle - Lifecycle Marketing Software

Don't fret over the logo yet

If you already have a logo or you can pull together one in a day, use it. Otherwise, skip it for now and use just your product name.

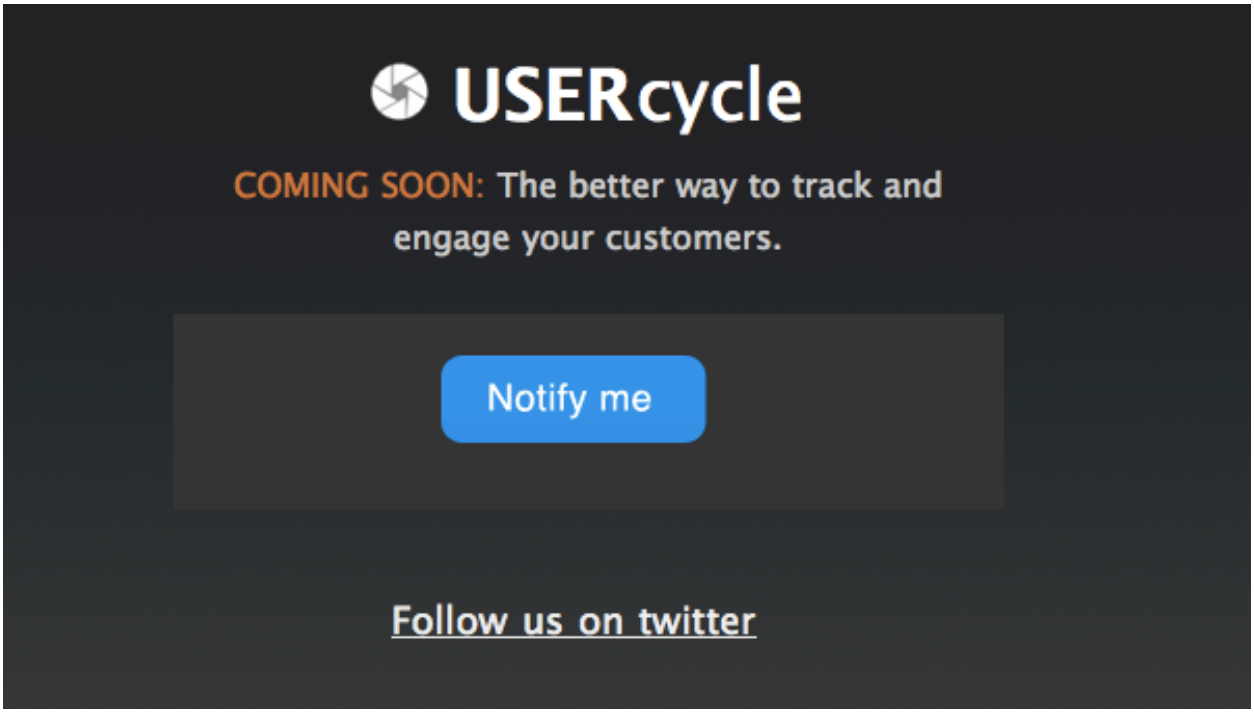
Collect email addresses

Pick your favorite tool, like CampaignMonitor, MailChimp, etc., to collect email addresses using a "Notify Me" call-to-action button.

Measure your website

Start with a FREE analytics tool like Google Analytics to track visitors on your landing page.

Here's an example of an early teaser landing page I used for USERcycle:



Note: The logo was a relic from my last product that I repurposed here. It will be "re-designed" at some point to be determined in the future.

Start a Blog

If you want to open a brownie store, become an "expert" in brownies first.
- Gary Vaynerchuck

Gary Vaynerchuck's book "Crush It" had a huge influence in getting me to hit the reset button on my previous blog and start over. I started writing about Lean Startups simply because I had more questions than answers. Writing really helped me crystallize my own thinking and grok different topics in the process. The side-effect of blogging led me to build a small but growing audience, meet "experts", write this book, and stumble into problems that led to 2 products: Lean Canvas and USERcycle.

Apart from "personal brand" benefits, blogging is also a great tool for discovering and driving customers to your product. As with the landing page, you need to start as early (and as simply) as possible:

Separate your personal brand from your product brand

If you have a personal blog, you can certainly use it to test your problem and recruit early prospects provided your audience matches your target demographic. But in the long run, it's better to separate the two and invest in building your product brand separately.

Use the product domain for your blog

To get the best bang from an SEO perspective, use your product domain for your blog. For example, http://blog.usercycle.com or http://www.usercycle.com/blog.

Pick a blogging platform

There are many low-cost options to start with: Wordpress, tumblr, posterous. My preference is Wordpress mainly because of the rich collection of 3rd party themes and plugins available. With Wordpress, you also have an upgrade path to services like WPEngine that help you scale later down the road.

Don't spend more than 5 minutes picking a theme

Previously I would have spent an afternoon or two designing a custom theme that matched my product's brand. DON'T waste your time. 3rd party themes are more than "good enough" to get you started. Once your blog is pulling its own weight, then worry about themes.

Write your first post - visualize your customer and problem

A good first post is to write about the top problem you're addressing. If you are your own prototypical customer, write from your experience. Otherwise, try and visualize your prototypical customer and write it from their point of view.

Keep it real

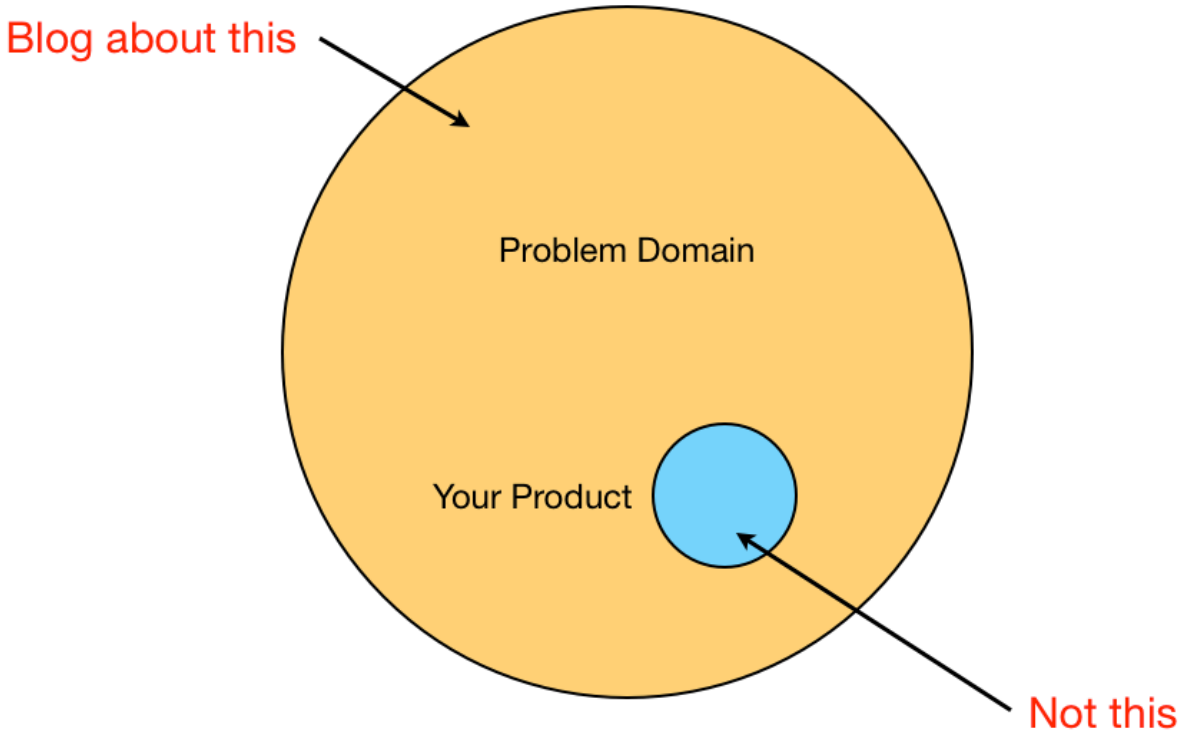
When it comes to writing, I don't rely on any SEO keyword tools or try and pack my posts with strategically placed keywords. It's important to have a voice and write naturally. A typical post runs between 500-1000 words which is a lot of room for the right "key" words to come out organically.

Measure your blog

In addition to tracking the number of unique visitors and page-views, also track comments and re-tweets.

Don't blog about your product

People can see right through self-promotional writing. The main purpose of your blog is to out-teach your competition and connect with customers. Rather than blogging about the product, go one or two levels higher and blog about the larger problem domain.



Write on a schedule

Writing is hard and takes discipline. Pick a weekly or bi-weekly schedule and stick with it.

Be patient

Most important of all, be patient. You might only get a few readers in the beginning but writing is an investment that keeps on giving. I frequently have posts I wrote over a year ago that all of a sudden see a spike of activity and get re-tweeted.

Alternatives to Blogging

If writing is not really your thing, find some other medium that works for you. The content, not the medium, is what matters.

Here are a few ideas that might help:

1. Video blog
Gary Vaynerchuck found it easier to talk to a camera than write so he created a set of short video blogs which was the start of Wine Library TV. He turned a New Jersey liquor store into a $50 million business 2 years later.

2. Tumblr / twitter
Hiten Shah, KISSmetrics, built a loyal following of several thousand followers by curating valuable links to startup related articles.

3. Create other types of content
Post an instructional video on YouTube, teach a workshop, write a book, speak at events, etc. The list goes on....

"A software company needs to act like it is a publishing company."

- Dharmesh Shah
Hubspot: Inbound Marketing

Find Prospects

Before I cover specific techniques for finding prospects, lets first review a few guidelines on interviewing:

How many prospects do I need?
I recommend preparing yourself to interview 30-60 people over a 4-6 week period which comes to talking to 2-3 customers a day with some time built-in for iteration. This should get you through both the Problem and Solution interviews.

Your goal is not statistical significance but finding strong signals for or against your hypotheses. The actual numbers could vary based on the strength of the signal you receive and your specific business model. As a general rule, I recommend speaking to no fewer than 10 people for the Problem Interview and 20 people for the Solution Interview.

Cast a wider net initially
Even though your first objective will be honing in on the defining attributes of early adopters, not all your prospects will (or should) be early adopters. It's actually better to start with a broader sweep of initial prospects at this stage and refine from there. You will have ample opportunity to narrow down your filter in the next round of interviews.

"Recruit loosely and grade on a curve."

- Steve Krug
Rocket Surgery Made Easy

Set an expectation of learning over pitching

People are generally willing to help if you set the right expectation of seeking their advice over trying to pitch to them. You'll see some examples of the kind of phrasing to use in the following section.

Don't pay prospects or provide other incentives

Unlike usability testing where it is acceptable to provide incentives for participation, your goal here is to find customers that will pay you, not the other way around.

Consider outsourcing interview scheduling

My least favorite part of interviews is scheduling them. You have to email people, coordinate around their schedules, juggle timezones, etc. If you do a little upfront work, you might be able to successfully delegate this task to someone else (like a virtual assistant).

Here's how I have made this work:

- I script all my email requests for interviews
- I clear my afternoons so it's easy to schedule interviews
- I'm copied in all the emails so I can intervene when needed.

How to Find Prospects

Whenever possible, you want to prioritize finding prospects through a channel you will actually use to acquire future customers. Unless you already have a path to customers, this may not be possible at this stage.

Here is a list of other techniques you can use to find and recruit interviewees:

1. Start with your 1 degree contacts
The first place to start is with your immediate contacts that meet your target customer demographic. Some are wary that feedback received from close contacts may be biased. My view is that **talking to anyone is better than talking to no one**.

2. Ask for introductions
The next step is asking your 1 degree contacts for introductions to people that meet your customer demographic. It's a good idea to include a message template that your contacts can simply cut-n-paste and forward to save them time.

Hey [friend] -

Hope all is well... I have a quick favor to ask.

I've got a product idea that I'm trying to validate with wedding photographers. My goal is to chat with local photographers to better understand their world and evaluate if it's worthwhile pursuing this product.

I'd really appreciate it if you could send this message along to people you know who fit this target.

(Feel free to change it a bit if you like):

Hello -

We are an Austin based software company and are currently working on a new service to simplify how photographers showcase and sell their images online. We are specifically building better and faster tools for online proofing, archiving, and selling.

I would love to get 30 minutes of your time to help us understand your current workflow. I'm not selling anything, just looking for advice.

Thanks,

Ash

3. Email list from teaser page

The teaser page we created earlier is a great source for finding people to interview. While you may not know whether they meet your target customer demographic, they represent people that were motivated enough to act on your UVP. Reach out to them and ask if they'd be willing to spend 20-30 mins with you on a call.

4. Cold Calling/Emailing/LinkedIn

Depending on your target demographic, you might be able to get a list of potential prospects that you could directly contact. Just be prepared for lower response rates which are par for the course when "cold calling".

Decouple the Problem From the Solution

"Customers don't care about your solution. They care about their problems."

- Dave McClure
500Startups

Dave McClure, 500Startups, has sat through hundreds of entrepreneur pitches and will probably sit through hundreds more. During these sessions, he has repeatedly called out entrepreneurs for spending a disproportionate amount of time talking about their solution and **not** nearly enough about the problem they are solving.

Having more passion for the solution than the problem is a problem.

Investors, and more importantly customers, identify with their problems and **don't care about your solution (yet)**. Entrepreneurs, on the other hand, are naturally wired to look for solutions. But chasing after solutions to problems no one cares enough about is a form of waste.

Building the right minimum viable product requires you to really grok the problem from the customer's point of view first. You do this by breaking up the customer interviews into two phases: the Problem Interview, and the Solution Interview.

The Problem Interview

Your main objective here is identifying your **early adopters** and learning how they **currently solve** these problems.

An early adopter is a customer who ranks one or more of the problems you're solving as a must-have and will generally pay to have it solved.

This information is invaluable in determining your positioning, pricing, and true **competition**.

Your competitors are NOT who you think they are BUT who your customers think they are.

The Solution Interview

Armed with a prioritized must-have problem list and an understanding of how early adopters address this problem today, you are then in the best position to define and test a solution. Even here, you don't jump to build the solution just yet. You build a "demo" first and use that to test the must-have features, positioning, and pricing.

The next 2 chapters will cover the Problem and Solution Interview process in detail along with sample interview scripts that you can repurpose for your own product.

CHAPTER 4

THE PROBLEM INTERVIEW

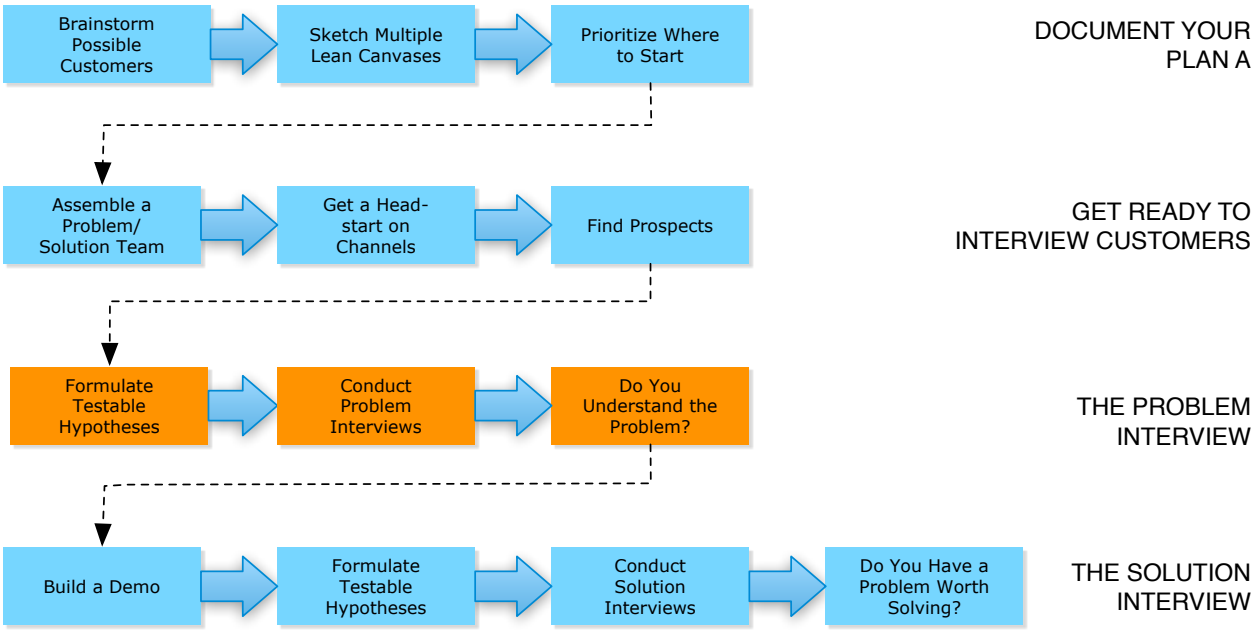
Talk to customers to understand their worldview before formulating a solution.

What You Need to Learn

Formulate Testable Hypotheses

Conduct Problem Interviews

Do You Understand the Problem?


Brainstorm Possible Customers
Sketch Multiple Lean Canvases
Prioritize Where to Start
DOCUMENT YOUR PLAN A
Assemble a Problem/ Solution Team
Get a Head-start on Channels
Find Prospects
GET READY TO INTERVIEW CUSTOMERS
Formulate Testable Hypotheses
Conduct Problem Interviews
Do You Understand the Problem?
THE PROBLEM INTERVIEW
Build a Demo
Formulate Testable Hypotheses
Conduct Solution Interviews
Do You Have a Problem Worth Solving?
THE SOLUTION INTERVIEW

What You Need to Learn

Before you can really define a solution, you have to really understand the problem.

The Problem Interview is all about validating your hypotheses around the "Problem-Customer" pair. Earlier you identified the top 3 problems you were addressing along with who you thought needed them solved the most.

In the Problem Interview you are specifically looking to answer the following questions:

Customer Segments: Who has the pain?

- *How to identify early-adopters?*

Problem: What are you solving?

- *How do customers rank the top 3 problems?*
- *What is their pain level: must-have, nice-to-have, don't-need?*
- *How do customers solve these problems today?*

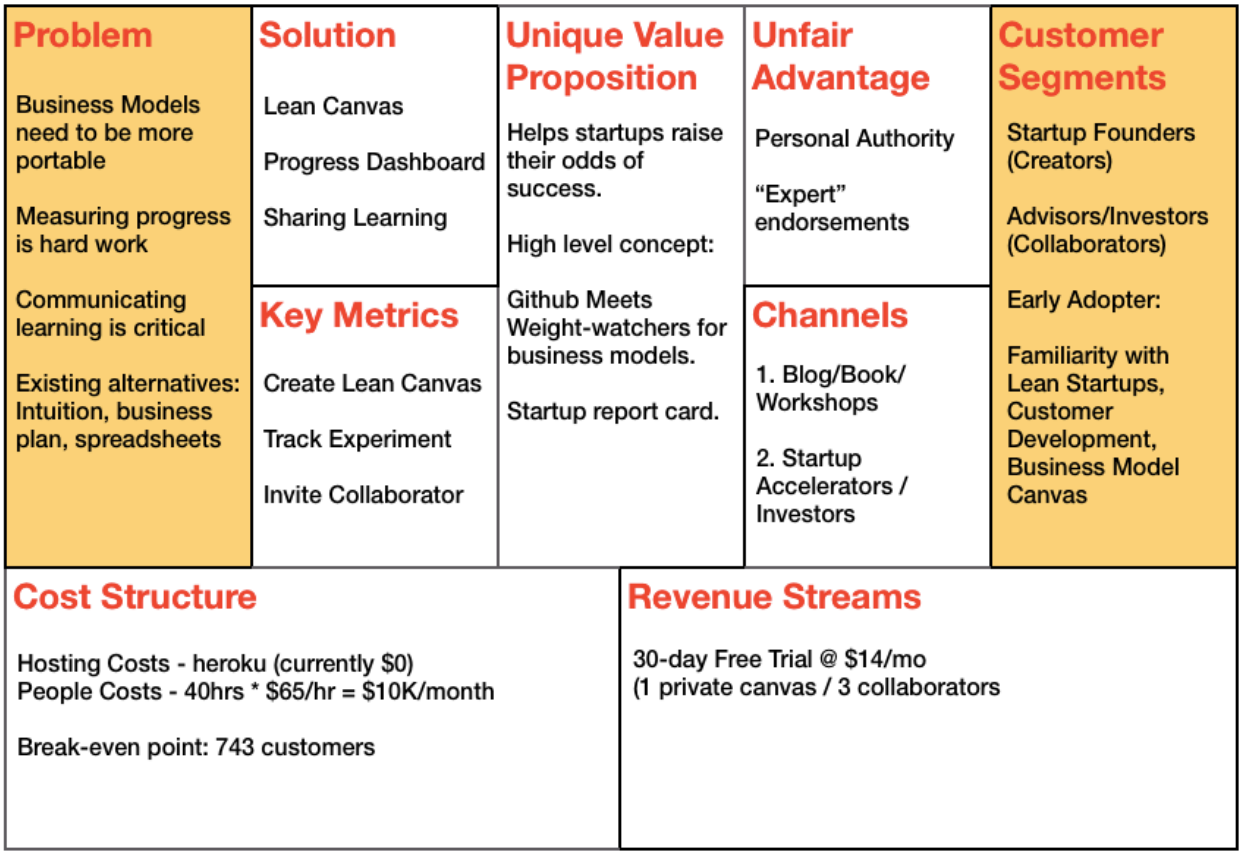
The Problem Interview process described here is a refinement of the "Problem Presentation" Steve Blank described in his book "The Four Steps to the Epiphany".

Formulate Testable Hypotheses

In order to make the interview results actionable, you need to take an additional step to convert the hypotheses from your canvas into testable hypotheses. This process is best illustrated with an example.

Case-Study: Lean Canvas

Here is my canvas from earlier with the sections under test highlighted:



Customer Segments

I articulate distinguishing characteristics of early adopters as hypotheses:

Hypothesis: Early adopters will be early-stage, bootstrapped startup founders
Hypothesis: Early adopters will be familiar with Lean Startup

Problem

While I believe documenting and measuring business model hypotheses can be a pain, there are "good enough" alternatives in the form of Power Point/Keynote templates, spreadsheets, business plans, etc. What I believe is harder to do is effectively communicate progress to internal and external stakeholders (advisors and investors).

Here's my expected ranking for the top 3 problems:

Hypothesis: Communicating progress will be voted as the #1 problem.
Hypothesis: Measuring progress will be voted as the #2 problem.
Hypothesis: Documenting hypotheses will be voted as the #3 problem.

On pain-level, there needs to be an implicit expectation that the #1 problem is a "must-have". Otherwise, it's not worth solving.

Hypothesis: Communicating progress will be voted as a must-have problem.

I'll further make a pain-level prediction for the other 2 problems:
Hypothesis: Measuring progress will be voted as a must-have problem.

Hypothesis: Documenting hypotheses will be voted as nice-to-have problem.

While there are many alternatives for documenting hypotheses: intuition, business plan, spreadsheets, I believe my early adopters most likely started with Steve Blank's worksheets from his book, and primarily use spreadsheets and email for measuring and communicating progress.

Hypothesis: Early adopters currently document their hypotheses using worksheets.

Hypothesis: Early adopters currently use spreadsheets to measure progress.

Hypothesis: Early adopters currently use email for communicating progress.

Conduct Problem Interviews

Here are some general notes on conducting a successful interview:

Prefer face-to-face interviews
I stressed the importance of being able to see your interviewees earlier. Other than picking up on body language cues, I find that meeting someone in person instills a sense of closeness that you can't recreate virtually. This is critical in customer relationship building.

Pick a neutral location
I prefer to run interviews in coffee shops to create a more casual atmosphere. Doing it at a prospect's office makes it more "business-like" and makes it feel more like a sales pitch - which it shouldn't be. That said, I'll agree to meet the prospect wherever they choose.

Ask for sufficient time
My interviews typically run between 20-30 mins without feeling rushed. Make sure you set the right time expectations upfront and are respectful of their time.

Stick to a script
There is a method to interviewing. In order to collect meaningful responses, it is important to maintain consistency in how you conduct interviews. It doesn't help, for instance, to tweak your story after every interview. Remember, this is not a pitch. At the same time, the script should also provide ample wiggle room so you can ask follow-up questions and explore new areas.

Conduct the interview with at least one other person

It always helps to have one other person in the room during the interview to make sure nothing slips through the cracks. But more importantly, it helps keep the learning objective.

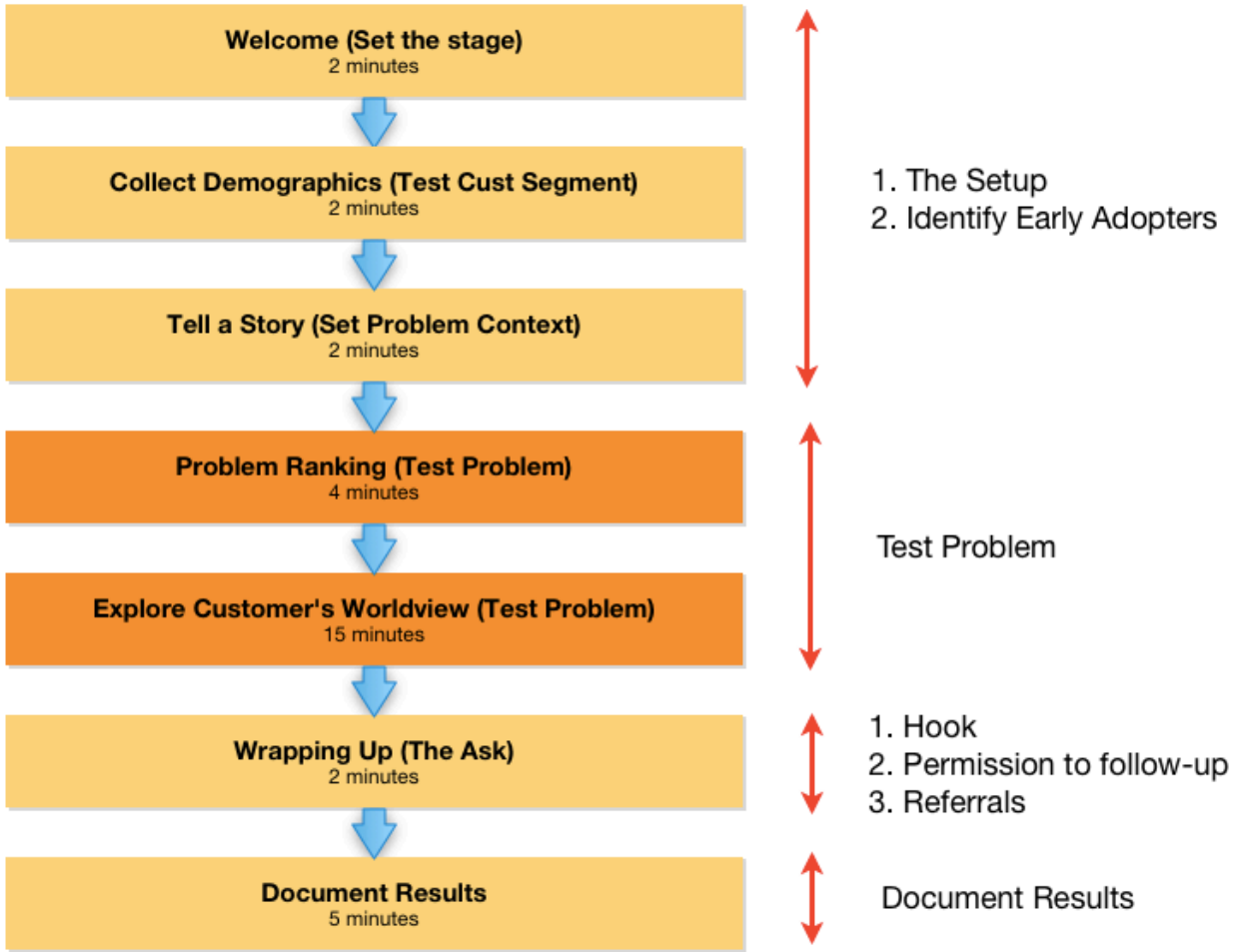
Avoid recording the interviewee

I tried recording interviewees early on (with their permission), but found that it made some people self-aware during the interview. That coupled with the fact that I never really went back to listen to an interview made it a non-starter for me. Your mileage may vary.

Document results immediately after the interview

I recommend spending 5 minutes immediately following an interview to document the results while your thoughts are fresh. Debrief with others later.

The Problem Interview Script Outline



Welcome (Set the stage)
2 minutes

Briefly set the stage for how the interview works.

Thank you very much for taking the time to speak with us today.

We are currently working on a tool that helps startups navigate the process of finding a business model that works. The idea for the tool came from my own experiences applying Lean Startup and Customer Development techniques to my last product.

But before getting too far ahead of ourselves, we wanted to pause and take the time to learn how other startups use business models today and see if there are real problems here worth solving.

The way the interview works is: I'll start by describing the main problems we are tackling and then I'll ask if any of those resonate with you.

I'd like to stress that we don't have a finished product yet and our objective is learning from you - not selling or pitching to you.

Does that sound good?

Collect Demographics (Test Customer Segment)
2 minutes

Ask some introductory questions to collect basic demographics that you believe will drive how you segment and qualify your early adopters.

> Before we go on to the problems, I'd like to ask you where you heard about Lean Canvas. Was it through my blog?
>
> Ok great. Can you also tell me a little about your product?
>
> Have you launched already?
>
> How long have you been working on it?
>
> Are you funded?
>
> Are you familiar with Lean Startups?
>
> Are you applying Lean Startup techniques to your product?

Tell a Story (Set Problem Context)
2 minutes

Illustrate the top 3 problems with a story.

Great, thanks. So let me tell you about the problems we are tackling:

The first problem we encountered was documenting our business model hypotheses in a format that was portable and easy to update. We knew it was important to write this stuff down and we got started by using the worksheets in Steve Blank's book. Like a business plan, these worksheets became difficult to track and maintain over time.

Also, once we had our plan documented, we didn't really have a good way for measuring progress. As an example, we interviewed a lot of customers and stored the results of the interviews in an internal wiki. But the process of going back weeks later to aggregate and make sense of these results was quite painful.

Another problem we constantly faced was effectively communicating progress to internal and external stakeholders like advisors and investors. We especially wanted a way to engage our advisors in more frequent tactical decisions while being respectful of their time.

Problem Ranking (Test Problem)
4 minutes

State the top 3 problems and ask for your prospect's ranking of them.

So to summarize:

The first problem was finding a portable format for documenting business model hypotheses.

The second problem was simplifying how we measure ongoing progress.

And the third problem was finding more effective ways to communicate progress to both internal and external stakeholders.

Do any of these problems resonate with you?

How would you rank these problems?

Have you run into any other problems I didn't talk about?

Explore Customer's Worldview (Test Solution)
15 minutes

This is the heart of the interview. The best script here is "no script".

Go through each problem in turn. Ask them how they address it today. Then sit back and listen.

Let them go into as much detail as they wish. Ask follow-up questions but don't lead them or try to convince them on the merits of a problem (or solution).

In addition to their raw responses, judge their body language and tone to get a sense of how they'd rate the problem: "must-have", "nice-to-have", or "don't need".

If they offer up new problems along the way, explore them the same way.

<Start with their #1 problem>

So, how do you deal with <problem> today?
Ask any follow-up questions to understand their current workflow.

<Repeat for other problems>

Wrapping Up (The Hook and Ask)
2 minutes

We are done with all of the hypotheses-related questions but you still have one more thing to do and two more things to ask them.

Even though you aren't ready to talk about your solution in detail, you need to provide a hook to maintain interest. The high-concept pitch is perfect for this. It not only helps explain your solution at a high-level, but leaves a memorable sound-bite which helps the interviewee spread your message.

Then you need to ask for permission to follow-up. Your goal is to establish a continuous feedback loop with prospects. And finally you need to ask the interviewee for referrals to other potential prospects.

> As I mentioned at the start, this isn't a finished product, but we are building an online solution to help startups document and systematically test their business models. The best way to describe the concept might be: "Github meets Weight-watchers" (or "The Startup Report Card").
>
> Based on what we talked about today, would you be willing to see the product when we have something ready?
>
> Also, we are looking to interview other people like yourself. Do you know any other startup founders applying Lean Startups techniques you think we could interview?

Document Results
5 minutes

Take the next 5 minutes immediately following an interview to document your results while they're still fresh in your mind.

It helps to create a template like the one on the following page so you can quickly jot down the responses to the hypotheses you set out to test.

I recommended earlier that you run the interview with one other person whenever possible to keep the results objective. Each of you should independently fill out the form first. Then have a debriefing session later where you compare notes and make a final entry into whatever "system" you use to record your interview results.

PROBLEM INTERVIEW Date: .....

Contact Info

Name: .....

Email: .....

Demographics

Title/Role: ..... Company: .....

Product: .....

Launched: ..... Funded: .....

Familiarity with LeanStartups: .....

Problem 1: Documenting Business Models

Priority Ranking: ..... Pain Level: .....

How Problem is addressed today: .....

.....

Problem 2: Measuring Progress

Priority Ranking: ..... Pain Level: .....

How Problem is addressed today: .....

.....

Problem 3: Communicating Progress

Priority Ranking: ..... Pain Level: .....

How Problem is addressed today: .....

.....

Notes: .....

Referrals: .....

Do You Understand the Problem?

In this section I'll discuss how to make sense of your interview results, refine the interview script, and determine when you are done.

Review your results weekly
If you are scheduling interviews at a good pace, you should be talking to 10-15 people a week. Don't change the script during the week. Rather, debrief at the end of each week to review that week's batch of interviews, summarize your learning, and make any adjustments (pivots) to your script.

The kinds of adjustments you make will vary based on the type of hypotheses you are testing and the strength of the signal you are getting from interviewees. The goal is to adjust the script and customer demographic along the way so you can incrementally get stronger and more consistent positive signals with each subsequent batch.

Start to hone in on early adopters
Look for identifying demographics among the responses that were most favorable i.e. strong problem resonance. Were these single-founder bootstrapped companies, funded companies, companies with advisors/ investors?

Similarly drop segments that were least favorable. For instance, if a particular segment doesn't see the value in business model testing and don't think they have a problem, while they may eventually make good latter stage customers, they are not going to make as good early adopters.

Refine the problems

If you get a strong “don’t need” signal across the board, drop that problem from the script. Similarly, if you discover a new “must-have” problem, add it to the script. Your eventual goal is to distill your product down to one “must-have” problem - one unique value proposition.

Really understand their existing alternatives

Understanding your early adopters’ existing alternatives is key to formulating the right product. Early adopters will use their existing alternatives as anchors against which they will judge your solution, pricing, and positioning. So for instance, if their existing alternatives are all free, your product has to promise and deliver enough value to overcome free.

Pay attention to words customers use

The best way to uncover the “key” words to use in your UVP is by listening closely to how customers describe their workflow.

Identify the potential paths to reaching early adopters

Once you start getting a sense of who the early adopter is, start identifying the path to reach more people like them. We’ll start testing these channels in the Solution Interview next.

What is the Problem Interview exit criteria?

You are done when you have interviewed at least 10 people and

- can identify the demographics of an early adopter,
- have a must-have problem and
- can describe how customers solve this problem today

CHAPTER 5

THE SOLUTION INTERVIEW

Test the solution with a “demo” before building the actual product.

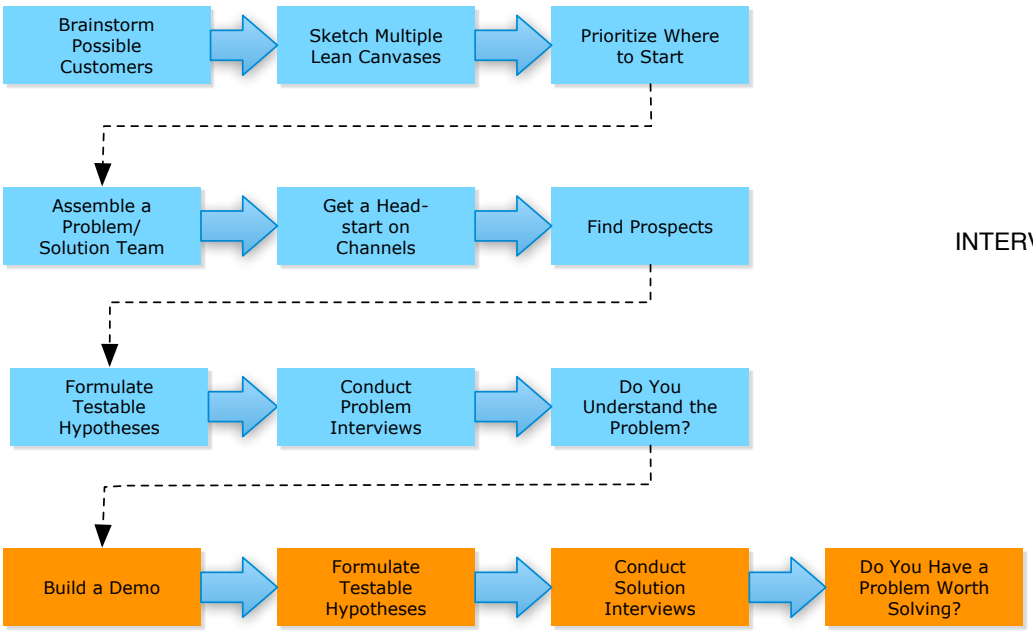
What You Need to Learn

Build a Demo

Formulate Testable Hypotheses

Conduct Solution Interviews

Do You Have a Problem Worth Solving?


Brainstorm Possible Customers
Sketch Multiple Lean Canvases
Prioritize Where to Start
DOCUMENT YOUR PLAN A
Assemble a Problem/ Solution Team
Get a Head-start on Channels
Find Prospects
GET READY TO INTERVIEW CUSTOMERS
Formulate Testable Hypotheses
Conduct Problem Interviews
Do You Understand the Problem?
THE PROBLEM INTERVIEW
Build a Demo
Formulate Testable Hypotheses
Conduct Solution Interviews
Do You Have a Problem Worth Solving?
THE SOLUTION INTERVIEW

What You Need to Learn

Armed with a prioritized problem list and refined customer definition, you are now ready to formulate a solution. In this section you will build a "demo" with which you will test for Problem/Solution Fit.

You will start by double-checking your learning from the Problem Interview:

Customer Segments: Who are your early-adopters?

- *Can you accurately describe your early-adopters?*

Problem: What is the problem you are solving?

- *What is the #1 must-have problem?*
- *How is it addressed today?*

Then, look to answer the following additional questions:

Solution: How will you solve these problems?

- *What is the minimum feature set needed to launch?*

Revenue Streams: What is the Pricing Model?

- *Will customers pay for a solution?*
- *What price will they bear?*

The Solution Interview process described here is a refinement of the "Product Presentation" Steve Blank described in his book: "The Four Steps to the Epiphany".

Build a Demo

I use the term “demo” loosely to refer to anything that can stand in place for the actual solution. The main objective is using the “demo” to illustrate visually how you intend to solve the problem, convey your unique value proposition, and test pricing.

Most customers are great at articulating problems, but not visualizing solutions.

For web applications, mockups are a great way to “demo” your solution. I use mockups quite heavily both before and after launch to qualitatively test features. While any mockup (hand-drawn, wireframes, photoshop, etc.) is better than no mockup, I have a strong preference for relying on mockups created using the “technology framework” in which the product will be delivered.

Here’s why:

The mockup needs to be realize-able
I have friends at design studios who have special teams in place just to build early user mockups/demos in Flash. These demos are very much part of the sales process and a lot of emphasis is placed on them. While they are quite effective at making the sale, they make the job of the implementation team quite difficult – with many of the more “flashy” elements being sometimes impossible to recreate in HTML/CSS. This leads to a disconnect in what is promised (and sold) to the client and what is eventually delivered.

Photoshop mockups, while not as “flashy”, can still be embellished with hard-to-recreate gradients, fonts, and shadow effects that can deviate enough from the actual user experience to make them hard to deliver too.

“We aren’t designing copies of web pages, we’re designing web pages.”
- Andy Clarke

The mockup needs to look real

I don’t like going the other extreme either of relying on barebones wireframes or sketches. While they are faster to put together, they require the customer to take a leap of faith on the finished product which I try and avoid.

The more real your “demo” looks, the more accurately you’ll be able to test your solution.

That said, given the choice, I will always prefer a wireframe or a sketch over a Flash demo.

The mockup needs to be quick to iterate

You will probably get valuable usability feedback during the interviews that you’ll quickly need to incorporate and test in subsequent interviews. This is where outsourcing your mockups to an external designer could actually hurt you if your ability to iterate is driven by their schedule.

The mockup needs to minimize on waste

Creating a mockup in anything other than the final technology in which the product is delivered creates some waste. In the case of a flash/photoshop mockup, the only thing that survives is the design. Everything else has to be redone into HTML/CSS.

How to build a demo

Build a skeleton app

Use your web application framework of choice (Ruby on Rails, PHP, etc.) to build a simple skeleton app. For those familiar with the Model-View-Controller pattern, I build my demos by going in the complete reverse order. I'll first setup some controllers, then sketch the views on paper which I'll then translate to HTML/CSS. The model is something I tackle only after I've qualitatively tested a screen and got a strong signal to implement it.

A mockup per problem

Create a single representative screenshot per problem and fill any gaps with narrative during the interview. The goal is not testing flows or usability but the content on your most critical screens.

Use real-looking data

Instead of using "dummy data" (lorem ipsum), come up with "real-looking" data that will not only help you lay out your screen but will support your solution narrative.

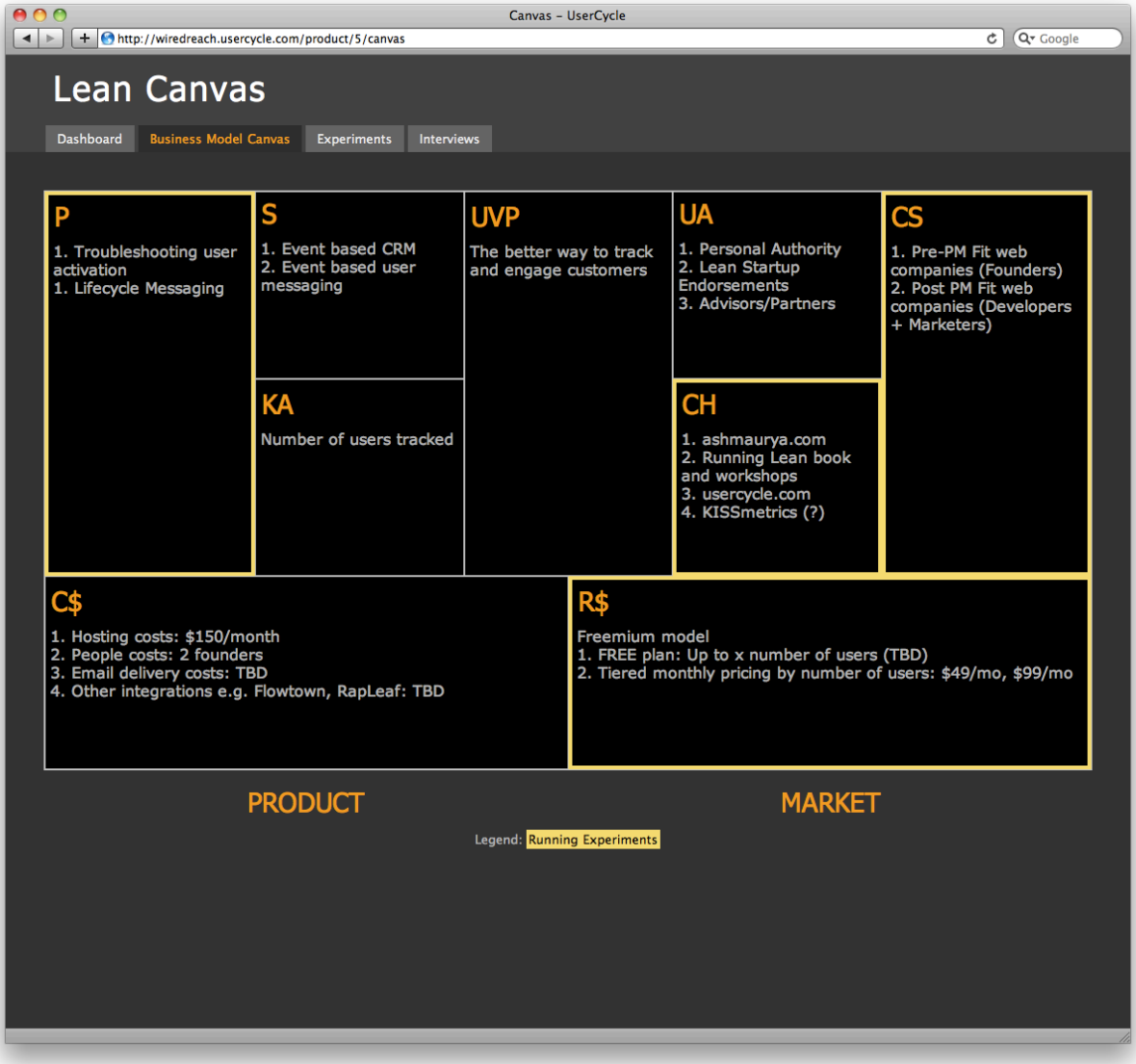
"Content precedes design. Design in the absence of content is not design, it's decoration."

- Jeffrey Zeldman
A List Apart

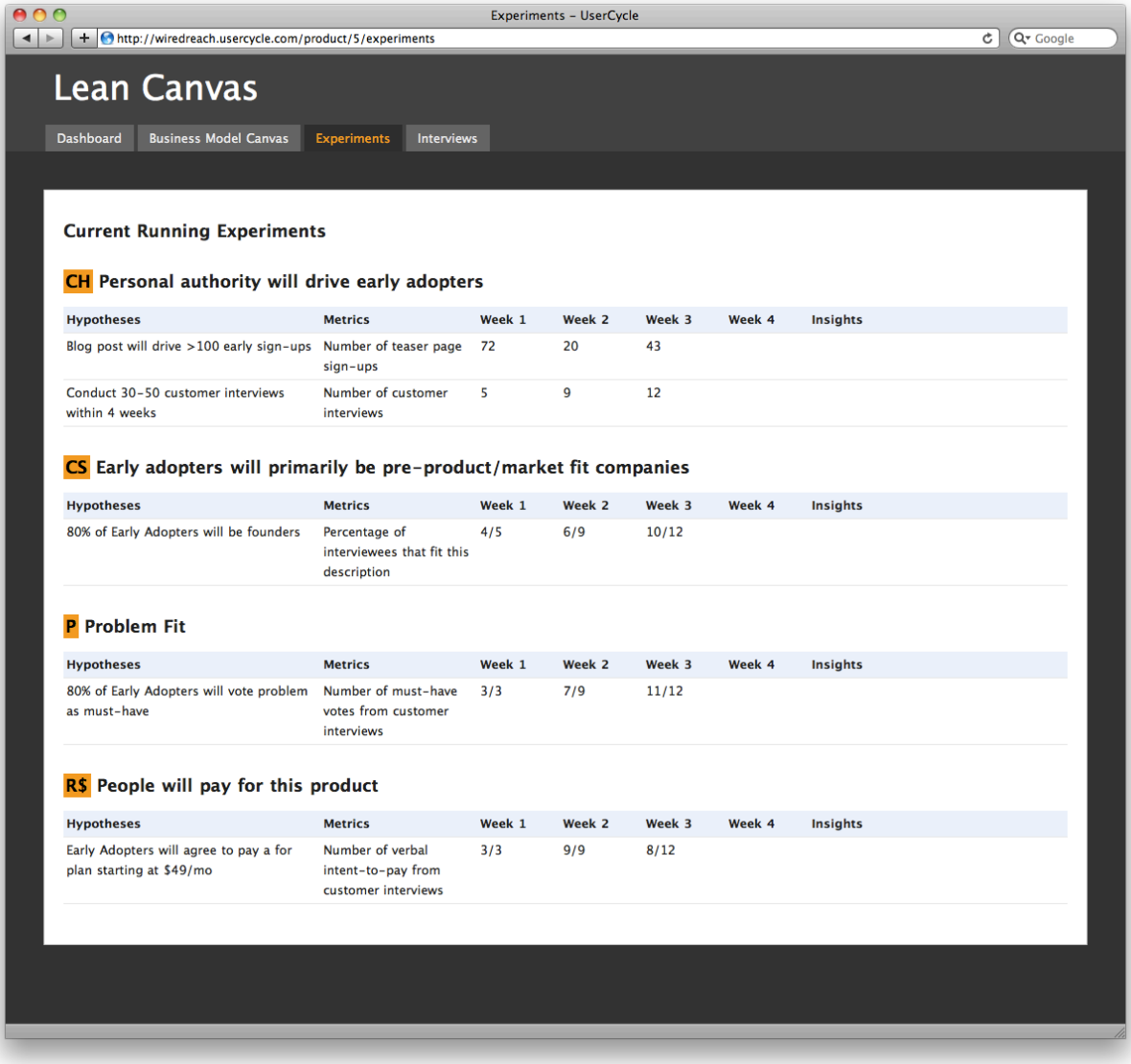
Mini Case-Study: Lean Canvas

For each of the problem/solution pairs I formulated earlier, here are the mockups I created:

Problem: Business Models need to be more portable
Solution: Use the Lean Canvas format for capturing hypotheses on 1 page



Problem: Measuring progress is hard work
Solution: Provide a simple way to "dashboard" experiments



Problem: Communicating learning is critical

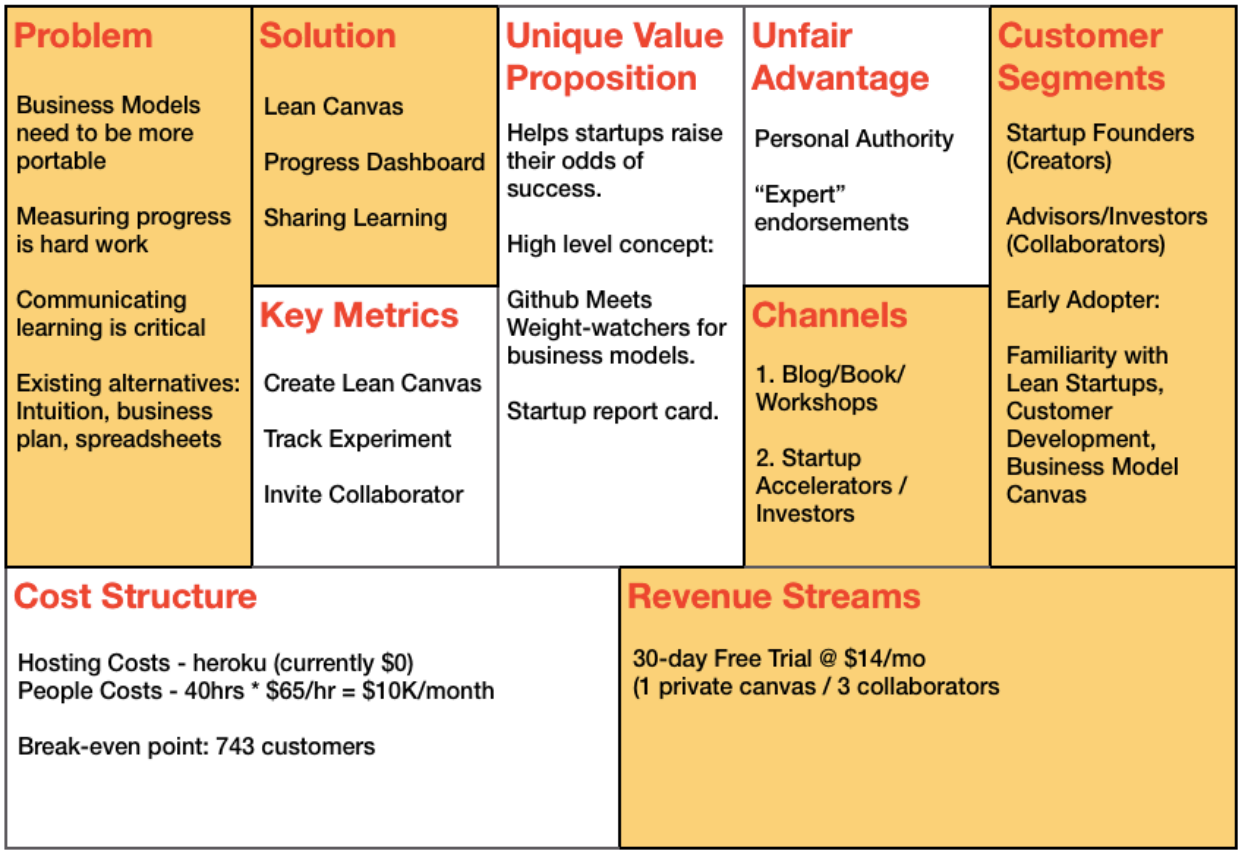
Solution: Need a sharing feature to facilitate collaboration

Formulate Testable Hypotheses

Once again, you need to document the testable hypotheses you intend to test during the interview. I'll illustrate with the Lean Canvas example:

Mini Case-Study: Lean Canvas

Here is our canvas from earlier with the sections under test highlighted:



We're going to test all the refined hypotheses from before, plus the following additional hypotheses on pricing and channels:

Solution

Hypothesis: The "Lean Canvas screen" will be a compelling solution for the "documenting hypotheses" problem.

Hypothesis: The "dashboard screen" will be a compelling solution for the "measuring progress" problem.

Hypothesis: The "lessons learned screen" will be a compelling solution for the "communicating progress" problem.

Revenue Streams

Hypothesis: Early adopters will agree to pay $14/mo for a 1 canvas/3 collaborator plan.

Channels

Hypothesis: Early adopters will find Lean Canvas via my blog, workshop, or book.

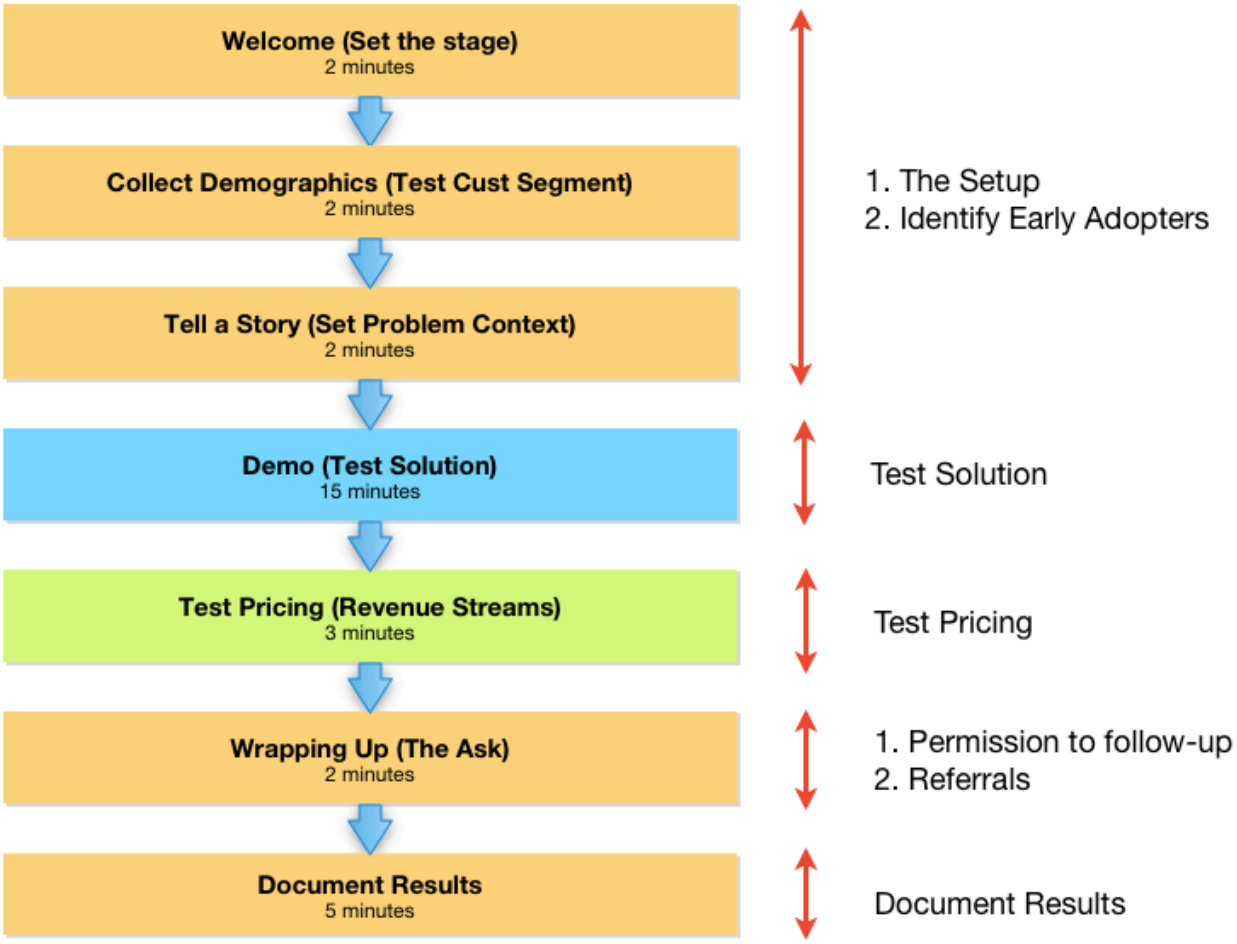
Conduct Solution Interviews

You're now ready to conduct the Solution Interview:

Use old prospects
You should have received permission to follow-up from your earlier Problem Interviews. If the prospect matches your early adopter demographic, arrange a follow-up Solution Interview with them.

Mix in some new prospects
It's a good idea to mix-in new prospects with every batch of interviews so you not only test all the hypotheses with a "beginner's mind". Your earlier interviews should have yielded some referrals which you could use. This is additionally the time to start testing any other channels you identified in your last iteration.

The Solution Interview Script Outline



Welcome (Set the stage)
2 minutes

As before, you need to briefly set the stage for how the interview works.

> Thank you very much for taking the time to speak with us today.
>
> We are currently working on a tool that helps startups navigate the process of finding a business model that works. The idea for the tool came from my own experiences applying Lean Startup and Customer Development techniques to my last product.
>
> The way the interview works is: I'll start by describing the main problems we are tackling and then I'll ask if any of those resonate with you. I also have some screenshots of the application. I'd appreciate your comments on them.
>
> I'd like to stress that we don't have a finished product yet and our objective is learning from you - not selling or pitching to you.
>
> Does that sound good?

Collect Demographics (Test Customer Segment)

2 minutes

Ask some introductory questions to collect basic demographics that you believe will drive how you segment and qualify your early adopters. If you've already interviewed this prospect, you can skip this section unless there are additional questions you've uncovered since you last interviewed them.

> Before we go on to the problems, I'd like to ask you where you heard about Lean Canvas. Was it through my blog?
>
> Ok great. Can you also tell me a little about your product?
>
> Have you launched already?
>
> How long have you been working on it?
>
> Are you funded?
>
> Are you familiar with Lean Startups?
>
> Are you applying Lean Startup techniques to your product?

Tell a Story (Set Problem Context)
2 minutes

As before, illustrate the top 3 problems with a story.

> Great, thanks. So let me tell you about the problems we are tackling:
>
> The first problem we encountered was documenting our business model hypotheses in a format that was portable and easy to update. We knew it was important to write this stuff down and we got started by using the worksheets in Steve Blank's book. Like a business plan, these worksheets became difficult to track and maintain over time.
>
> Also, once we had our plan documented, we didn't really have a good way for measuring progress. As an example, we interviewed a lot of customers and stored the results of the interviews in an internal wiki. But the process of going back weeks later to aggregate and make sense of these results was quite painful.
>
> Another problem we constantly faced was effectively communicating progress to internal and external stakeholders like advisors and investors. We especially wanted a way to engage our advisors in more frequent tactical decisions without overwhelming them and being respectful of their time.
>
> Do any of these problems resonate with you?

If you don't sense a strong problem resonance, don't continue with the Solution Interview, but rather use the Problem Interview script to learn more about how they solve these problems today.

Demo (Test Solution)
15 minutes

This is the heart of the interview.

Go through each problem in turn and illustrate how you solve it using the supporting mockup.

<For each problem>
  Illustrate how you solve the problem using the supporting mockup.
  Pause after each one and ask if they have any questions.
<Repeat for other problems>

So that's what the application looks like right now. We are trying to prioritize what to finish and release first and would like to ask you a few more questions.

Which of the screenshots resonated with you the most?

Which could you live without?

Are there any additional features you think are missing?

Test Pricing (Revenue Streams)
3 minutes

Finding the right price is more art than science.

Usually the right price is one the customer accepts but with a little resistance.

Test Pricing using the "starting price" you determined earlier for this Customer Segment. Your earlier probing on demographics should have revealed what price to test.

Gauge the customer's response immediately after you tell them the price. If they accept the pricing, make a note of whether they hesitated or readily accepted.

If you believe there are high switching costs or barriers to adoption, consider using Steve Blank's method of asking if they would use a solution if it were "Free" before testing a price.

> So lets talk about pricing next.
>
> We are envisioning launching the tool using a subscription model which will allow you to create a single canvas and invite up to 3 collaborators.
>
> Would you pay $14 a month to use a tool like this?

Wrapping Up (The Ask)
2 minutes

We are done with all of the hypotheses questions but you still have two more things to ask them.

The first is permission to follow-up to trial the service when it's ready. The second is referrals to other people you could potentially interview.

> Thanks a lot for your time today. You have been very helpful.
>
> As I mentioned at the start, this isn't a finished product but we are close to launching something soon. Would you be interested in trying out the product when we have something ready?
>
> Also, we are looking to interview more people like yourself. Do you know any other startup founders applying Lean Startups techniques we could interview?

Document Results
5 minutes

Take the next 5 minutes immediately following an interview to document your results while it's still fresh in your mind.

It helps to create a template like the one on the following page so you can quickly jot down the responses to the hypotheses you set out to test.

As before, have each interviewer independently fill out the form first. Then have a debriefing session later where you compare notes and make a final entry into whatever "system" you use to record your interview results.

SOLUTION INTERVIEW Date: .....

Contact Info

Name: .....

Email: .....

Demographics

Title/Role: ..... Company: .....

Product: .....

Launched: ..... Funded: .....

Familiarity with LeanStartups: .....

Solution Mockup 1: Documenting Business Models

Priority Ranking: ..... Pain Level: .....

Additional Comments: .....

Solution Mockup 2: Measuring Progress

Priority Ranking: ..... Pain Level: .....

Additional Comments: .....

Solution Mockup 3: Communicating Progress

Priority Ranking: ..... Pain Level: .....

Additional Comments: .....

Pricing

Will use if free: .....

Willing to pay ($X/month): .....

Notes: .....

Referrals: .....

Do You Have a Problem Worth Solving?

In this section I'll discuss how to make sense of your interview results, refine the interview script, and determine when you are done.

Review your results weekly
As before, wait till you have a week's worth of interviews to change the script.

Add/kill features
If you received specific usability or feature enhancements, discuss whether there were compelling reasons to incorporate them. Remove unnecessary features.

Confirm your earlier hypotheses
If you ended your Problem Interview iteration with strong positive signals, there should be no surprises here. Otherwise, revisit your older hypotheses and refine them until you get consistent results.

Refine pricing
If you got no resistance to your pricing, consider testing a higher price. Take the customers' alternative solutions into account. If their current solutions are Free, how can you provide more value to justify them paying?

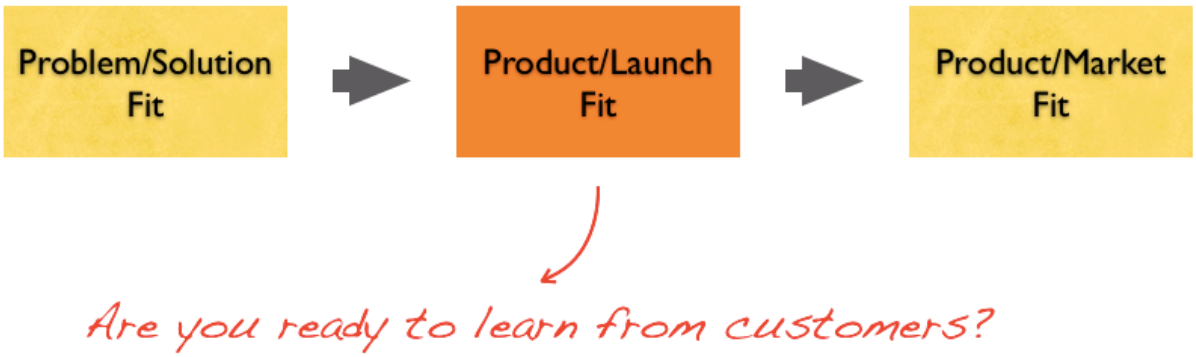
Again look for patterns. What was the price sensitivity of bootstrapped companies versus funded companies? Who is the prototypical early adopter and what price will they bear? Can you build a viable business at that price?

What is the Solution Interview exit criteria?

You are done when you can confidently

- identify the demographics of an early adopter,
- have a must-have problem,
- can define the minimum features needed to solve this problem, and
- have a price the customer is willing to pay
- that you can build a business around (using a back-of-the-envelope calculation).

PART 2: PRODUCT/LAUNCH FIT

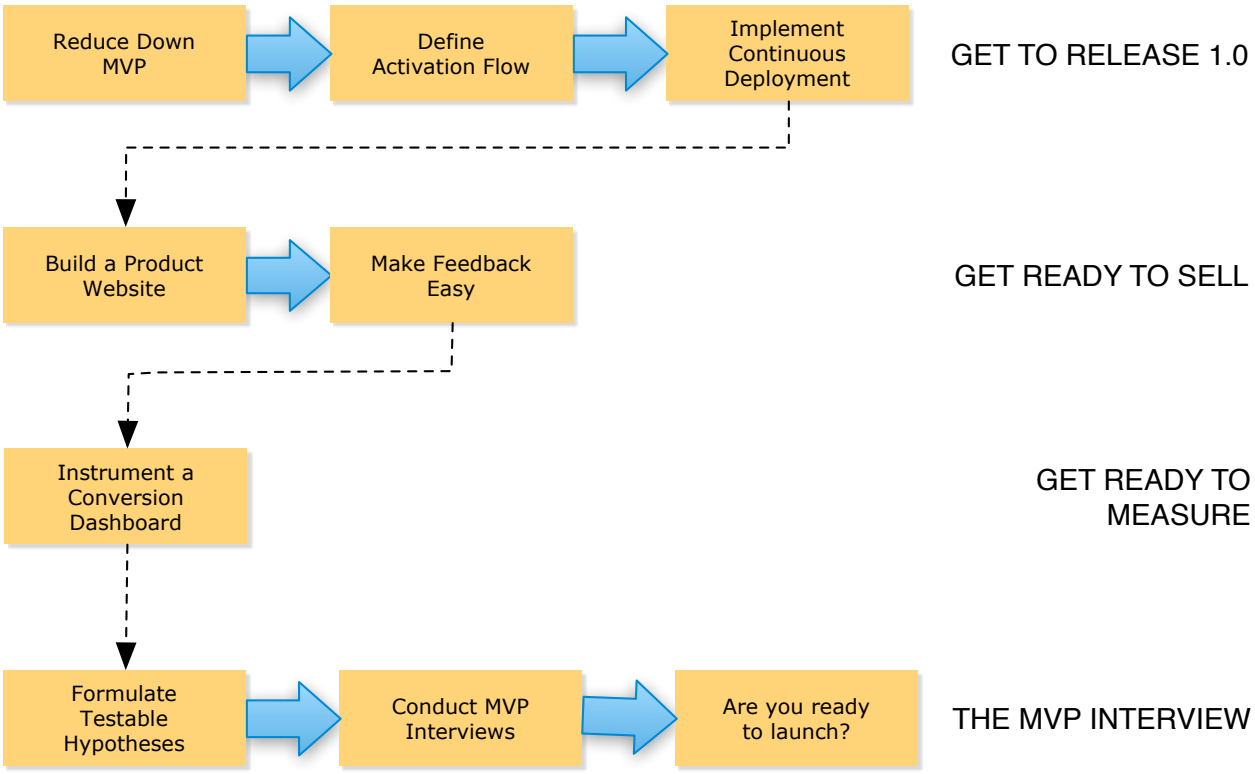


SUMMARY

Product/Launch Fit, in one page.

While you definitely understand your customers' needs better than just a few weeks ago, don't rush into building and launching your MVP just yet. It is fairly easy to get distracted during this stage and either build too much or build the wrong product.

You not only need to distill down your MVP to it's essence but also lay the groundwork to maximize your future iterations for **speed, learning, and focus.**



CHAPTER 6

GET TO RELEASE 1.0

There is very little learning from customers that happens during development and QA. Reduce scope and shorten the cycle time between requirements and release so you get to learning parts faster.

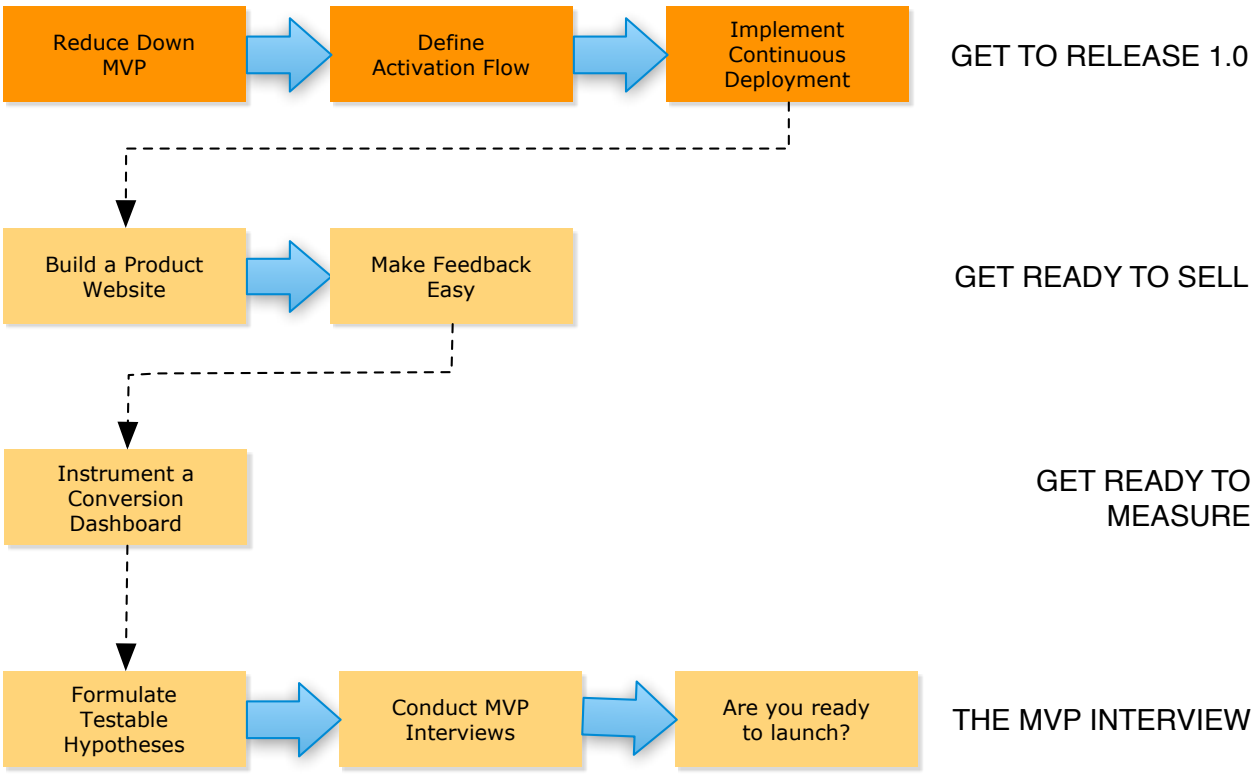
Product Development Gets in the Way of Learning

Reduce Down Your MVP

Understand the User Lifecycle

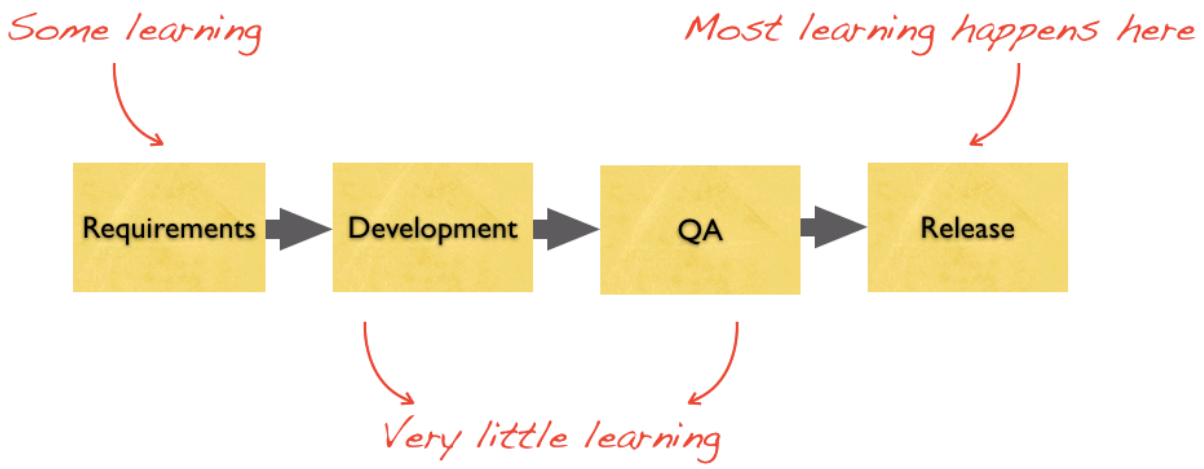
Define Your Activation Flow

Get Started Deploying Continuously


Reduce Down MVP
Define Activation Flow
Implement Continuous Deployment
GET TO RELEASE 1.0
Build a Product Website
Make Feedback Easy
GET READY TO SELL
Instrument a Conversion Dashboard
GET READY TO MEASURE
Formulate Testable Hypotheses
Conduct MVP Interviews
Are you ready to launch?
THE MVP INTERVIEW

Product Development Gets in the Way of Learning

Lets start by taking a closer look at where learning happens during a typical product development cycle:



While some learning happens during the requirements gathering stage, most learning happens after you release your product. Even though building a product is the purpose of a startup, very little learning happens during development and QA. Sure you're learning about other things then - just not about customers.

We obviously can't eliminate development and QA, but we can shorten the cycle time from requirements to release so you **get to the learning parts faster**.

The first step is reducing the scope of your MVP down to it's essence so you build the smallest thing possible.

Reduce Down Your MVP

A danger with iterating through mockups during the Solution interview is that it is quite easy to get carried away and end up with more than you need for your MVP. In order to reduce waste and speed up learning, you need to pare down your mockups so all you have left is the essence of your product - your Minimum Viable Product.

Reducing down the scope of your MVP not only shortens your development cycle, but also removes unnecessary distractions that dilute your product's messaging.

Your MVP should be like a great reduction sauce - concentrated, intense and flavorful.

Here's how to do that:

1. Clear your slate

Don't automatically assume any features have to be included in your MVP. Start with a clean slate and justify the addition of each one.

2. Start with your #1 Problem

The job of your UVP is making a compelling promise.
The job of the MVP is delivering on that promise.

The essence of your MVP should be captured in your #1 problem mockup. Start there.

3. Eliminate nice-to-haves and don't-needs

From your Solution Interviews, you should be able to label every element on your mockup as "must-have", "nice-to-have", or "don't-need".

Immediately eliminate the don't-needs, and add your nice-to-haves to your features backlog queue **unless** it is a prerequisite feature for a must-have feature.

4. Repeat Step 3 for your #2 and #3 Problem mockups

5. Consider other customer feature requests

Your customers may have highlighted certain features that are needed to make your product complete or usable, for example, Salesforce integration. Visit these next and add/defer them based on the "must-have" level of need.

6. Charge from day 1, but collect on day 30

It is a given for products nowadays to have some sort of a trial period. It is also generally a good practice to defer upfront collection of credit card information to reduce on signup friction and avoid negative option billing[2]. Both of these work to your advantage to further reduce scope. You don't need to worry about merchant accounts, recurring subscription providers, or supporting multiple plans for launch. You'll have 30 days after launch to get these things working.

7. Focus on learning, not optimization

All your energy needs to be channeled towards accelerating learning. Speed is key. Don't waste any effort trying to optimize your servers, code, database, etc. for the future. Chances are quite high that you **will not** have a scaling problem when you launch. In the rare event you do (a great problem), most scaling problems can be initially patched with additional hardware which you can justify because you should be charging your customers - buying you time to address the problem more efficiently.

---

2 Negative Option billing refers to a business practice where a merchant provides a free trial or sample of goods/services, requires a credit card upon sign up and then bills the customer in the future unless they proactively cancel with the merchant.

Understand the User Lifecycle

While **building something people want** is the ultimate goal, you can't do that unless you really understand your users.

When you first launch a product lots of things can and do go wrong. Many of these things aren't even about what we traditionally think of as "the product". For instance, you might be losing people due to poor positioning, design, or pricing.

Well guess what – design, positioning, and pricing are all parts of the product.

But tackling the whole product all at once can be overwhelming. This is where an understanding of the User Lifecycle is helpful in prioritizing what you focus on first and build your product in an iterative manner. It also helps you identify the key metrics to measure and keep your learning actionable.

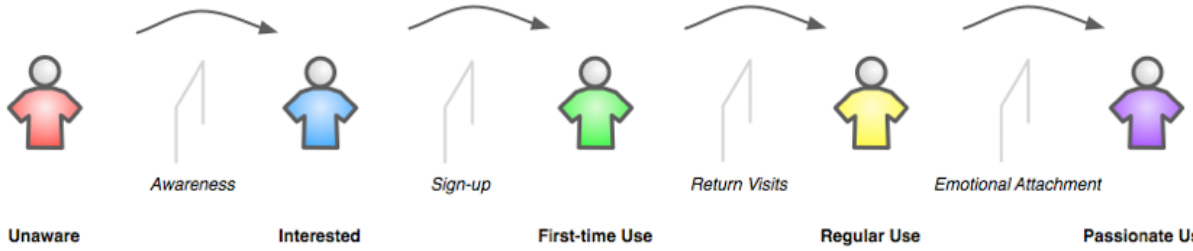
What is the User Lifecycle?

The user lifecycle describes the path a user takes from first landing on your website to eventually becoming a passionate user.

It helps to consider two models to understand the user lifecycle:

Model 1: The Usage Lifecycle

This model was built by Joshua Porter, Performable, and captures the user lifecycle **qualitatively** from the perspective of the user.



I particularly like Joshua's model because it focuses on understanding your users' motivation and psychological hurdles by stage which is key to proper design and positioning.

Pirate Startup Metrics (AARRR)

The one shortcoming of the "Usage Lifecycle" is that it doesn't tell you how to measure each stage.

To answer that, it's helpful to consider another model, built by Dave McClure, that captures the user lifecycle **quantitatively** using five key metrics:

These five metrics form the basis for defining a "macro" conversion funnel for your product.

A macro metric is a roll-up of several steps or sub-funnels.

Note: While the startup metrics were created with web-based products in mind, they are "basic enough" to support many other kinds of products and businesses.

Here's a brief description of each "macro" metric:

Acquisition
Acquisition describes the path a customer takes from first landing on your website as an unaware visitor to becoming an interested prospect. You measure interest by tracking what they do on your website. If they leave your website without clicking anywhere (abandonment), you weren't able to generate interest. For instance, I measure successful acquisition as getting my visitors to view my sign-up page.

Activation
Once you have an interested customer, Activation describes the path a customer takes from signing up to your service to having a gratifying first user experience.

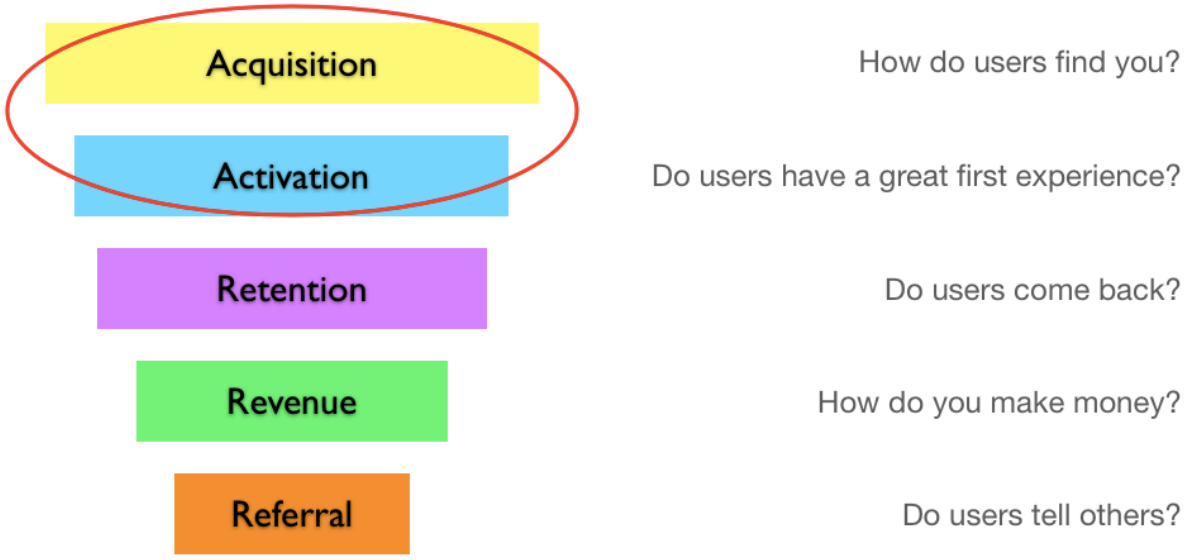
Retention
Retention measures “repeated-use” or engagement with your product. As we’ll see in part 3 of the book, this is one of the key metrics to measuring Product/Market Fit.

Revenue
Revenue measures the events that get you paid. Since most products have (or should have) a trial period, the revenue events typically occur after Activation and some repeated-use - Retention.

Referral
Referral is a more advanced form of a user acquisition channel where your happy customers refer or drive potential prospects into your conversion funnel. This can range from implicit viral or social sharing features (like Share with a friend) to explicit affiliate referral programs or Net Promoter Score.

Focus on Launch

It's fairly easy to expend a lot of effort defining and instrumenting your complete conversion funnel which is a form of waste at this stage. While all five metrics are needed to capture the full conversion funnel, you only need to focus on the first two metrics for launch: Acquisition and Activation.



Acquisition gets new prospects into your funnel and Activation gets them to start using your MVP. That is our goal for Product/Launch Fit.

Let's start with Activation.

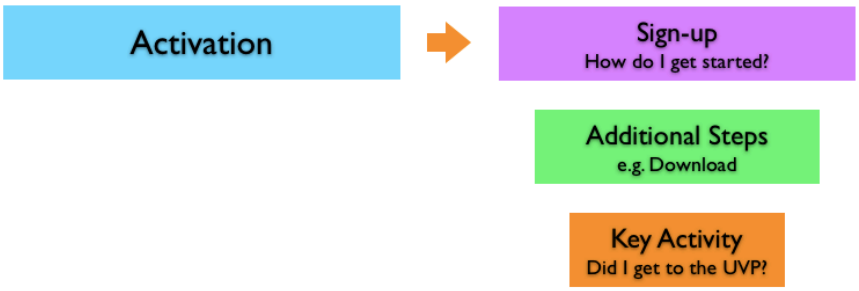
Define Your Activation Flow

Once you have distilled down your features list, you are ready to start defining your Activation Flow:

Your Activation Flow describes the path customers take from signing-up for your service to having a gratifying first experience.

The Anatomy of an Activation Flow

The Activation flow is a sub-funnel made up of the following steps:



While the ultimate objective of your activation flow is getting your customers to experience your unique value proposition as quickly as possible, most of what goes wrong right after you launch happens here.

For this reason, it is far more critical to architect your activation flow for learning over optimization.

Here are some ways to do that:

Reduce signup friction but not at the expense of learning
It is generally a good practice to keep your signup forms short and only collect what you absolutely need but don't shy away from asking for critical contact information (like an email address) upfront.

"Forms are the least of our problems."

- Joshua Porter
Bokardo

Case-study: CloudFire
CloudFire is a downloadable desktop application and in the interest of simplifying the signup process, we put up a simple "Download" button on the home page and deferred the account creation step to post-installation.

Our analytics showed a huge discrepancy between the number of downloads and signups. We knew we were loosing people during the installation step but didn't know why. At first, we tried implementing several hypotheses (best guesses) such as reducing the installer size, supporting an online installer, etc. which took several weeks to implement but with no noticeable impact. Then we moved up the sign-up screen and started asking for an email address before the download step.

While this didn't fix the activation problem, it allowed us to identify the users that ran into these problems and reach out to them. Several of them responded back which helped us quickly uncover a few critical issues we hadn't caught before that were the key to solving our low activation rates.

Reduce the number of steps but not at the expense of learning

The same principle of architecting for learning over optimization also applies to the number of steps in your activation flow. While it is important to reduce on the number of steps, it is far more important to keep critical steps separate so you can troubleshoot where people drop off when things go wrong.

Case-study: posterous (blogging platform)

An extreme example that comes to mind is the posterous landing page when they first launched. Rather than asking you to sign-up for an account, they asked you to email them a message with the contents of your first post. While this was a novel idea, they were essentially asking you to "abandon" their landing page to go send an email which treated an activated user the same way as an uninterested visitor. This flow, while highly optimized, offered zero opportunity for learning when things went wrong.

Deliver on your Unique Value Proposition

A good activation flow needs to deliver on the promise established on your landing page. When you map out your activation flow make sure it demonstrates your UVP - preferably in one sitting. You only get one chance to make a good first impression.

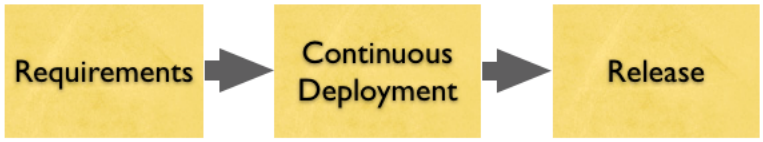
Be prepared for when things go wrong

Offer inline troubleshooting and provide multiple ways customers can reach out for help - email, 1-800 number, etc.

Get Started Deploying Continuously
(Mostly for developers)

Another technique for shortening the cycle time from requirements to release is implementing a continuous deployment process.

Continuous Deployment is a practice of releasing software continuously throughout the day - in minutes versus days, weeks, or months.



Of all the Lean Startup techniques, Continuous Deployment is one the most controversial. One of the immediate concerns usually raised is around quality - comparing Continuous Deployment to “cowboy coding”.

Implemented correctly, Continuous Deployment does **not** shortcut on quality and actually demands much stricter testing and monitoring standards. Continuous Deployment is not just practiced by small startups but in use at larger companies like IMVU, Flickr, and Digg. These companies collectively serve millions of users a day and have built fairly sophisticated continuous deployment systems to ensure high quality standards which leads to the second concern.

The second concern usually raised is that building such a continuous deployment system is a massive and daunting undertaking. It is. But these systems were built incrementally over a span of years. As we’ll see, the

Continuous Deployment process is itself a feedback loop for continuous learning and improvement which lends itself well to starting out small and why I am covering it now.

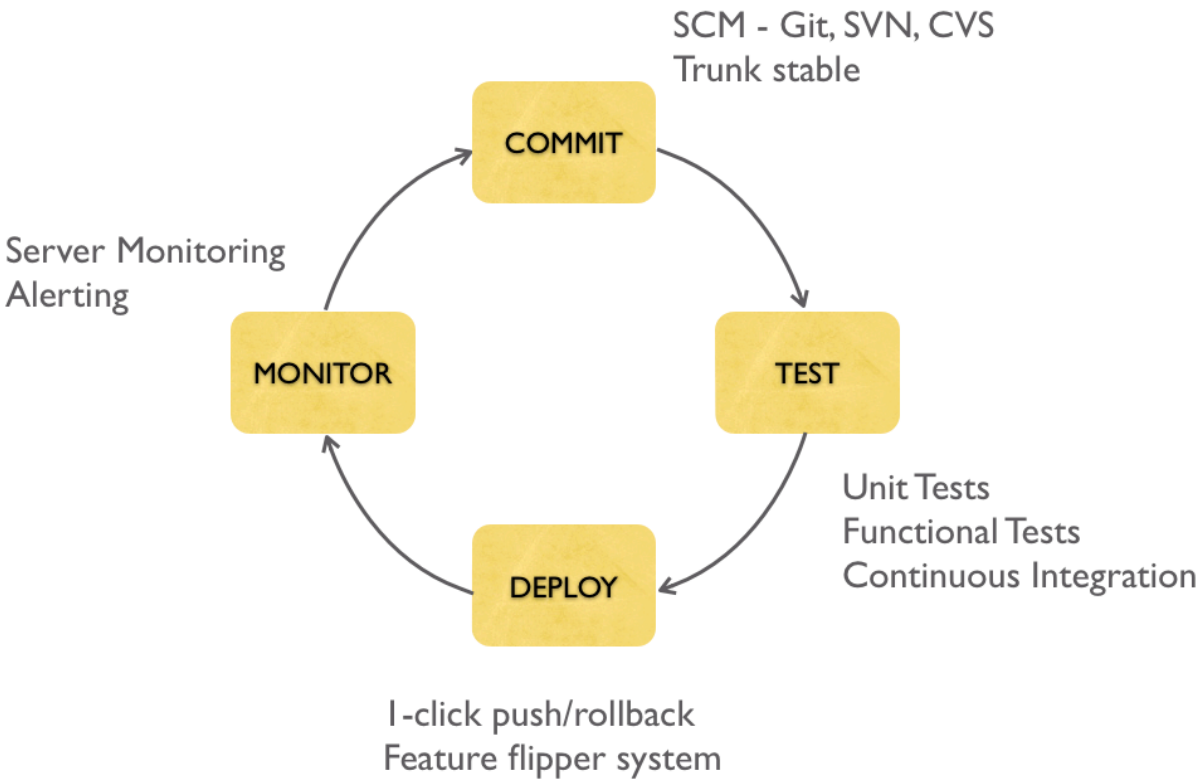
Right now is the perfect time to lay the groundwork and practice with Continuous Deployment - while you don't have customers, code or servers, to worry about. While Continuous Deployment won't help you launch your MVP faster, starting with a basic system won't slow you down and will help lay the foundation for speeding up future iterations after you launch.

It is also important to point out that while Continuous Deployment deploys code into production in small batches, this code does **not** have to live for your users. There is a distinction between a "software release" and a "marketing release".

Continuous Deployment is a big topic - maybe worthy of it's own book. Rather than getting bogged down with a lot of theory, I'd like to focus on the key takeaway of **speeding up** your learning loops and present what you need to get started.

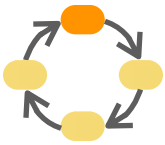
How to Get Started

Here's an overview of the continuous deployment cycle:



You probably already have (or should have) the pieces needed to put together a basic continuous deployment system.

I'll cover each each stage of the continuous deployment cycle next.



COMMIT

One of the ways Continuous Deployment strives to reduce waste is by reducing work-in-progress inventory i.e. un-deployed code. Having lots of un-deployed code increases inertia and reduces your ability to react quickly (more integration, more co-ordination, more planning).

Here are two techniques that help you reduce your work-in-progress inventory:

1. Code in smaller batches

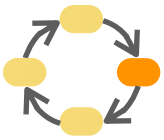
The basic idea here is to deploy less code but more frequently. The definition of a small batch is relative but strive to make it **as small as possible**. I used to deploy code on a bi-weekly schedule with my last product and eventually got my batch size down to the output of a two hour coding session. Sure you won't usually finish a full feature in two hours but you get pretty good at building and deploying features incrementally.

The number of lines of code in my average batch went from several hundred lines to about twenty five lines. A direct side effect of deploying less than twenty five lines of code versus hundreds is that troubleshooting unexpected production issues immediately after a deployment becomes a whole lot easier, as does fixing and releasing them.

2. Always be trunk stable

Another practice for reducing work-in-progress inventory is not using any branching in your source control tree. I know this sounds a little extreme because branching and merging are among the most touted features in a source control management system. They allow you to isolate big risky changes off the "stable" mainline trunk. But the longer you stay off the trunk, the more integration debt you collect which again inevitably leads to more integration risk, co-ordination, and planning headaches.

It's more efficient to train yourself to always be trunk stable and build and deploy your features incrementally. It's important to point out that deploying features incrementally doesn't necessarily mean they are made live to all users immediately. This allows you to incrementally roll out big features and make them available to select users like your internal team until it is ready to go live. I'll cover how you do this using a feature flipping system in the "Deploy" section.



TEST

Taking the Continuous Deployment plunge is particularly scary because it eliminates manual testing (QA) which typically has served as a safety net for catching defects after development and before deployment.

1. Testing is everyone's responsibility
First, I don't know any 2-3 person startup that has a QA department which makes testing everyone's responsibility. Second, having long test cycles creates the same work-in-progress inventory problems we discussed earlier. The solution is not creating a separate QA function but rolling it into the development process and investing more heavily in automated testing.

2. Use a continuous integration server
Set up a continuous integration server, like Hudson, to automatically trigger a build (if you have compiled code) and run your application against your test suite after every commit.

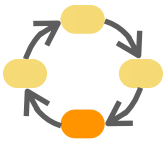
3. Do not tolerate any failing tests
I've worked in places before where developers train themselves to ignore failing tests because they know they are outdated. In a Continuous Deployment system, these tests serve as your last line of defense before pushing code into production and you cannot tolerate a single failing test especially since your ultimate goal is automating deployment.

4. Prefer functional tests over unit tests

I don't advocate achieving "full test coverage". On the contrary, I believe writing unit tests for obscure edge cases or creating complicated mocks is a form of waste and not the most optimal use of time when the focus is speed and learning. I instead prefer creating functional tests over unit tests whenever possible. There are several great options like Selenium and Sauce Labs for writing and automating functional tests for web applications.

5. Start with your activation flow

Building tests for features that your customers never get to is also a form of waste. When building tests, use your user lifecycle to prioritize your tests. Start with the activation flow and then incrementally add other functional tests over time.



DEPLOY

The deployment step pushes your tested code into your production environment. As this can get quite sophisticated when you have a cluster of machines, it is best to start early when you have a just a few servers:

1. Outsource as much of your server infrastructure as possible
Spending effort setting up and configuring your own servers at this stage is waste. You should instead pick a cloud or platform provider (like Amazon, or heroku) and focus all your efforts on building your application - not your infrastructure.

Note: Many cloud providers offer free tiers to get you started.

2. Create a separate staging area if you are so inclined
A separate staging area serves as an additional safety net before pushing code to production and can be a good idea for building up confidence in your deployment system. However, I've found staging areas of limited use beyond basic spot checking and at some point your continuous integration server should be able to serve this function in a more repeatable and automated fashion.

3. One-click push and rollback
The next step is writing a set of deploy scripts that can push your code to your production server and rollback your code to the last release. The rollback is used in the event you push a bad change. If you are deploying small enough batches, you shouldn't ever need to rollback beyond the last release.

Note: If you are on heroku, one click push and rollback is offered out-of-the-box.

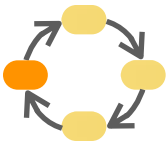
4. Deploy manually first, then automate

It is usually a good idea to run the push script manually at first and audit every deployment while you build up your confidence. If you are using Hudson as your continuous integration, it is fairly easy to add a task to automatically trigger your push script when you are ready for that.

5. Implement a simple feature flipper system

You will inevitably be faced with having to deploy a new "big" feature while maintaining the old and need a mechanism in place to isolate your users from these changes. A feature flipper system fits that bill.

A feature flipper system uses flags in your code that allows you to enable/disable features on a per user basis.



MONITOR

The job of your monitoring system is to be able to automatically detect, alert, and eventually even automatically recover from unexpected errors. An example of recovering from an error might be automatically triggering your rollback script in the event of a bad release. To be able to do this, your monitoring system would have to get pretty sophisticated and not just monitor your server health but also your application health i.e. business metrics.

The good news is that you don't need to start there. It is actually a form of waste to try and overbuild your monitoring system because Pareto's Principle rules here:

The Pareto Principle: Roughly 80% of the effects come from 20% of the causes.

The Continuous Deployment cycle has a built-in feedback loop that helps you build this monitoring incrementally.

Here's how to get started:

1. Start with out-of-the-shelf monitoring
There are numerous off-the-shelf monitoring and alerting applications like ganglia, Nagios, New Relic, etc. that you can use to start monitoring basic server health metrics.

2. Tolerate unexpected problems only once
The way you build up your monitoring system is by implementing a Five Whys root cause analysis to every unexpected problem you encounter.

*The **Five Whys** is a questions-asking method used to explore the cause/effect relationships underlying a particular problem. Ultimately, the goal of applying the Five Whys method is to determine a root cause of a defect or problem.*

The following example demonstrates the basic process:

My car will not start. (the problem)

Why? - The battery is dead. (first why)
Why? - The alternator is not functioning. (second why)
Why? - The alternator belt has broken. (third why)
Why? - The alternator belt was well beyond its useful service life and has never been replaced. (fourth why)
Why? - I have not been maintaining my car according to the recommended service schedule. (fifth why, a root cause)
Why? - Replacement parts are not available because of the extreme age of my vehicle. (sixth why, optional footnote)

I will start maintaining my car according to the recommended service schedule. (solution)

Source: Wikipedia

Applied to unexpected problems in your production environment, the outcome of each Five Why's analysis should provide a slew of tests, monitoring, and alerts that you then add to your existing suite.

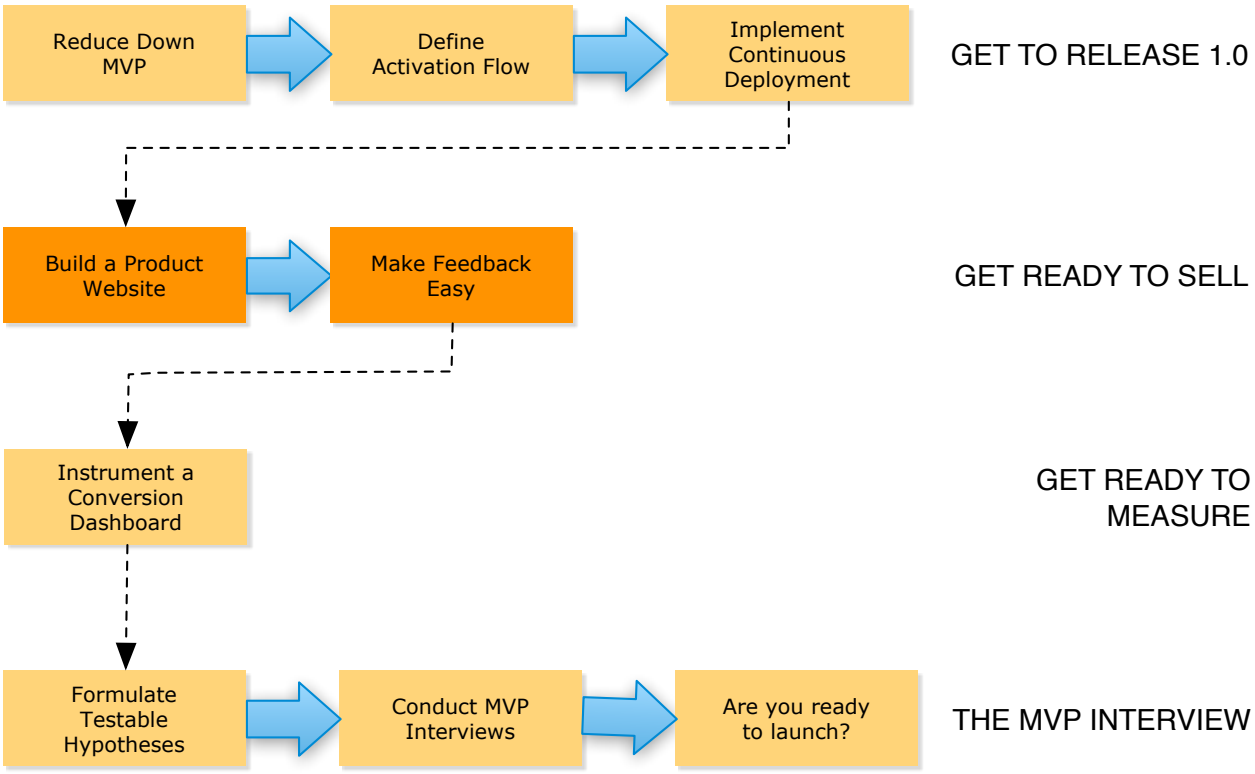
CHAPTER 7

GET READY TO SELL

A great product eventually sells itself, but it first needs to be sold which is the job of your marketing website. Your marketing website converts unaware visitors into interested prospects. The basic elements of a marketing website are your landing page and your pricing page.

Build a Product Website

Make Feedback Easy


Reduce Down MVP
Define Activation Flow
Implement Continuous Deployment
GET TO RELEASE 1.0
Build a Product Website
Make Feedback Easy
GET READY TO SELL
Instrument a Conversion Dashboard
GET READY TO MEASURE
Formulate Testable Hypotheses
Conduct MVP Interviews
Are you ready to launch?
THE MVP INTERVIEW

Build a Product Website

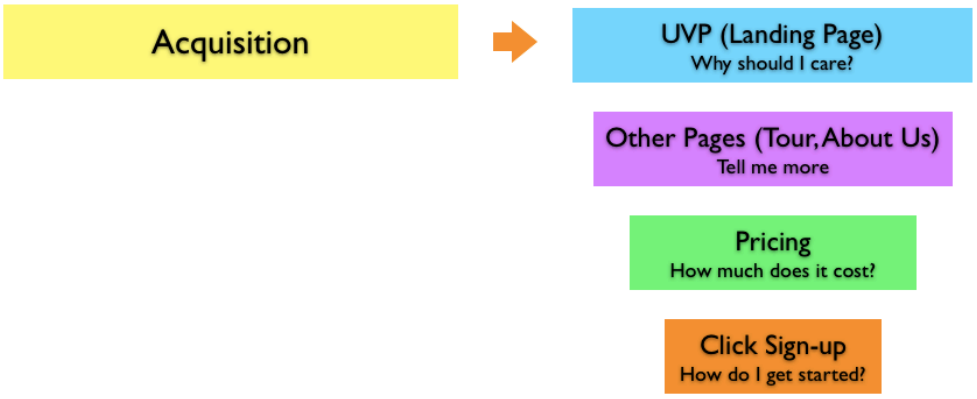
With your MVP built, it's time now to replace your teaser page with a full-fledged marketing website. The purpose of your marketing website is simple - **to sell your product**.

Your marketing website is critical in driving the Acquisition trigger in your user lifecycle.

Acquisition describes the path customers take from first landing on your website as an unaware visitor to becoming an interested prospect.

The Anatomy of a Marketing Website

Acquisition is itself a sub-funnel:



Following the principle of architecting for learning over optimization, I recommend starting with explicit pages for each step. Each page should have a primary call-to-action and a secondary call-to-action. My primary call-to-action directs visitors to my pricing page (Acquisition sub-goal)

while my secondary call-to-action offers a link to more information (e.g. product tour).

The landing page is by far the hardest of the three. It's job is to make a case for your product to an unaware visitor in less than 8 seconds. We'll deconstruct the elements of a good landing page next but I'll first list out a few other pages you probably should also include:

About Page
While the job of your Landing page is to provide a compelling reason to buy your product, the job of your About page is to provide a compelling reason to buy from your company. This is your opportunity to put a face to your product, to tell your story, and connect with your customers.

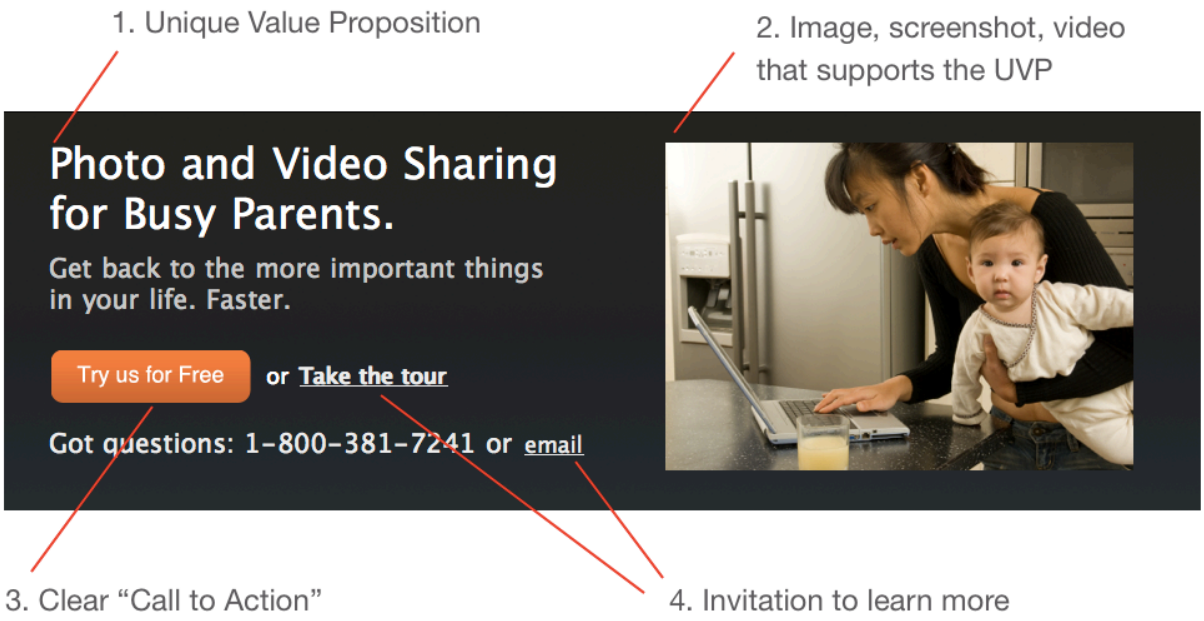
"Terms of Service" and "Privacy Policy
Both these pages are basic requirements for offering a service on the web. They are also fairly standard with lots of good examples online to model after.

Tour Page - Video/Screenshots
I usually defer this page to later and start with just the Landing Page. But if your customers are more analytical or research-oriented, you might need to provide a separate page with more details, technical specifications, etc.

It comes down to fundamentally understanding your customers and their motivations.

The Landing Page Deconstructed

While the Landing Page has the daunting task of very quickly connecting with your visitors, there are several basic elements that make up a successful landing page.



1. Unique Value Proposition

Put the latest refinement of your Unique Value Proposition here. This is the most important element of the page and needs to convey a unique and compelling reason to buy.

2. Supporting visual

Support your unique value proposition with a visual aid that resonates strongly with your target audience. The actual medium may be an image, a screenshot, or video depending on your specific audience.

3. A Clear "Call to Action"

Every page needs to have single clear "call to action". It should stand out and set a clear expectation as to what happens next.

4. Invitation to learn more

Some visitors may need more information before they're convinced. Provide additional links to your tour page (if you have one), or your 1-800 number (more on that in the next section).

The landing page shown above is missing one critical element:

5. Social Proof

Social proof elements help to raise your credibility and trust. They are typically provided through customer testimonials and "As Seen On" logos. The reason they are absent from the landing page above is that you most likely don't have these yet.

Make Feedback Easy

The fastest way to learn from customers is by talking to them.

Much like I favor interviewing customers over surveys, I prefer getting feedback from customers in person or over the phone than through other means like email, forums, or discussion boards.

Here's why:

It shows you care
A toll free number sends a signal to your users that you care and went the extra mile to make it easy for your customers to call you.

You don't have a scaling problem yet
Contrary to popular belief, you won't be bombarded by phone calls. A lot of my calls are typically from prospects with questions about the service than support issues. It is fairly easy to set calling hours during the day and re-route the phone if and when you run into a scaling problem (which is a great problem to have).

Tech Support is a continual learning feedback loop
After each call, I review the reason for the call and see if I can change something on the site - messaging, help, tips, pricing clarifications, etc. to continually improve the product.

Tech Support is Customer Development

Not only does talking to a customer help you better understand customer problems, but it provides you an opportunity to ask your customers a question or two.

Tech Support is Marketing

The opportunity for learning from your customers this way is so great that I have my mobile phone tied to the 800 number of all my products. Don't believe me. Call: 1-800-381-7241.

Having the founder of the company answer the phone further shows your commitment to listening to customers and I've found it empowers customers to open up even more.

Provide Multiple Feedback Channels

Even though I favor talking to customers, you need to provide multiple ways for customers to get in touch with you - email, twitter, blog, inline feedback form.

Avoid Voter-based Feedback Tools

I'm not a fan of voter-based tools like GetSatisfaction or User Voice because I don't believe all customers are equal. Listening to the most vocal or popular feedback does not guarantee you'll uncover the right learning to build a better product. More often than not, it has the exact opposite effect.

CHAPTER 8

GET READY TO MEASURE

You not only need to be able to visualize your user lifecycle but need to be able to measure it.

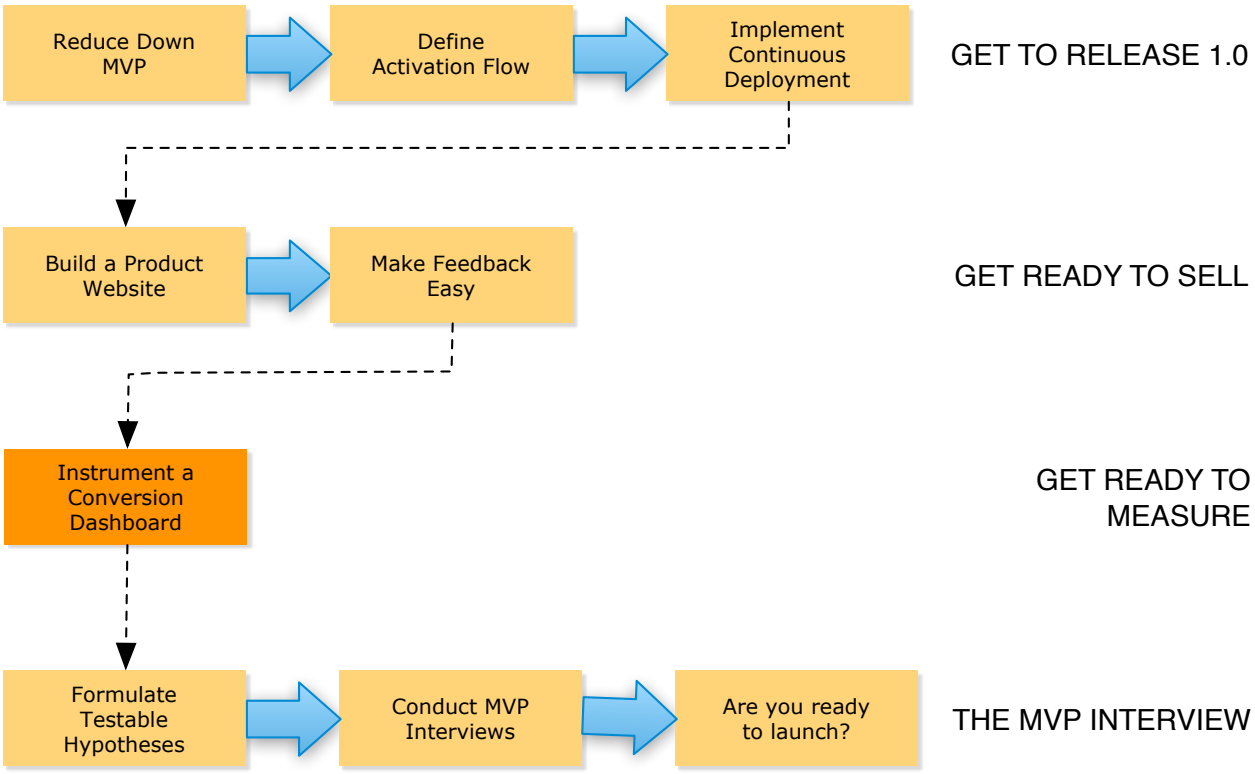
The Need for Actionable Metrics

Metrics Are People First

Funnel Reports Aren't Enough

Say Hello to the Cohort

How to Build Your Conversion Dashboard


Reduce Down MVP
Define Activation Flow
Implement Continuous Deployment
GET TO RELEASE 1.0
Build a Product Website
Make Feedback Easy
GET READY TO SELL
Instrument a Conversion Dashboard
GET READY TO MEASURE
Formulate Testable Hypotheses
Conduct MVP Interviews
Are you ready to launch?
THE MVP INTERVIEW

The Need for Actionable Metrics

Even though the terrain before Product/Market Fit is riddled with qualitative learning, you still need actionable metrics to be able to visualize and measure your user lifecycle.

The objective before Product/Market Fit is **not** as much about optimizing for conversion and **all** about quickly identifying and troubleshooting hot spots in your user lifecycle.

Up until now, you have made a number of product decisions based on what customers have told you, it's time to start measuring what they do.

What is an Actionable Metric?

An actionable metric is one that ties specific and repeatable actions to observed results.

The opposite of actionable metrics are vanity metrics (like web hits or number of downloads) which only serve to document the current state of the product but offer no insight (by themselves) into how you got there or what to do next.

Put another way, things like web hits or downloads, are elements of sub-funnels that make up the larger macro metric that matters, such as Acquisition and Activation.

It's not what you measure but how

Understanding the difference between a vanity metric and a macro metric is the first step. In order to make your metrics actionable, you have to additionally make them accessible (through simple reports), and auditable (be able to go behind the numbers).

The 3 AAAs of metrics are: Actionable, Accessible, and Auditable.

- Eric Ries
StartupLessonsLearned

I'll go into some detail how you do that in the next few sections and then I'll outline the steps for building a conversion dashboard.

Metrics Are People First

Eric Ries popularized the meme of "Metrics are People too" for the purpose of making your metrics auditable, but I don't believe it goes far enough.

While I am a big proponent of building a metrics driven culture, there is a lot more to building a great product than numbers. For starters, **you have to be able to go to the people behind the numbers**.

The ideal conversion dashboard is part analytics part customer relationship management.

Here's why:

Metrics can't explain themselves
When you first launch a product or new feature, lots of things can and do go wrong. Metrics can help you identify where things are going wrong, but they can't tell you why. You need to talk to people for that.

Don't expect your users to come to you
When a user first uses your product, they aren't yet invested in your solution. They usually start out interested but skeptical and their motivation decays fast when things go wrong. In other words, you can't expect them to promptly send in a bug report or pick up the phone and call you when they need help. They might, but it's more likely that they'll simply abandon your product and leave. The burden of quickly identifying problems and reaching out to your users is yours.

Not all metrics are equal

You've been very selective about who you've interviewed up until now. Once you launch, you won't be able to control who uses your product. In addition to your target early adopters, you might be visited by bots, curious onlookers, and maybe even other yet undiscovered target customers. When you just look at numbers, you get an averaging effect that can be greatly skewed if you don't yet have a lot of traffic (or the right traffic). You need a way to segment your metrics into different buckets by demographics.

Simple Funnel Reports Aren't Enough

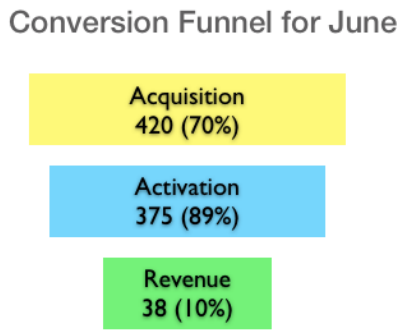
The Funnel Report is a powerful analysis tool. It's simple to understand and lends itself well to visually depicting a conversion dashboard. But most third-party implementations of funnel reports are better suited at tracking micro-level funnels, like landing page conversions, than macro-level funnels, like your user lifecycle.

Micro-level funnels are characterized by short lifecycle events typically measured in minutes, while macro-level funnels are characterized by long lifecycle events typically measured in days or months.

Simple funnel reports work by letting you specify a reporting period over which the number of key event occurrences are counted and visualized. This approach doesn't work when the intervals between events fall outside the reporting period.

To illustrate these problems, lets consider an example for a downloadable product that uses a 14-day trial.

Here's an example of what a typical funnel report might look like:



In the example above, the “Acquisition” and “Activation” events are all short lifecycle events while the “Revenue” event is a long lifecycle event.

This poses the following issues:

Inaccurate Conversion Rates
The numbers reported for the Revenue event most likely include purchases made in May and exclude purchases made in July which skews the overall conversion rates.

Dealing with traffic fluctuations
This skewing of numbers is further exacerbated by any fluctuations in traffic. If sign-ups go down in July, your conversion rates will appear to be better when they may not be.

Measuring progress (or not)
Another problem with this sort of reporting is that your product is also constantly changing. It is hard, if not outright impossible, to tie back observed results (good or bad) back to actions you took in the past such as launching a new feature.

Segmenting funnels
Over time you will probably run a split-test or need to segment your funnel to isolate one group of customers from another. You can’t do this with a simple funnel report.

Say Hello to the Cohort

So while funnels are a great visualization tool, funnels alone are not enough. The answer is coupling funnels with cohorts.

Cohort Analysis is very popular in medicine where it is used to study the long term effects of drugs and vaccines:

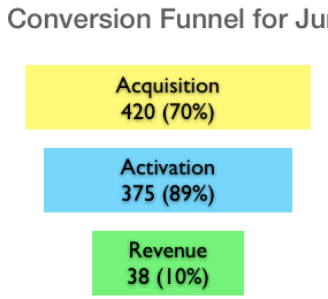
> *A cohort is a group of people who share a common characteristic or experience within a defined period (e.g., are born, are exposed to a drug or a vaccine, etc.). Thus a group of people who were born on a day or in a particular period, say 1948, form a birth cohort. The comparison group may be the general population from which the cohort is drawn, or it may be another cohort of persons thought to have had little or no exposure to the substance under investigation, but otherwise similar. Alternatively, subgroups within the cohort may be compared with each other.*
>
> Source: Wikipedia

We can apply the same concept of the cohort or group to users and track their lifecycle over time. For our purposes, a cohort is any property that can be attributed to a user. The most common cohort used is “join date” but as we’ll see this could just as easily be the user’s “plan type”, “operating system”, ‘gender”, etc.

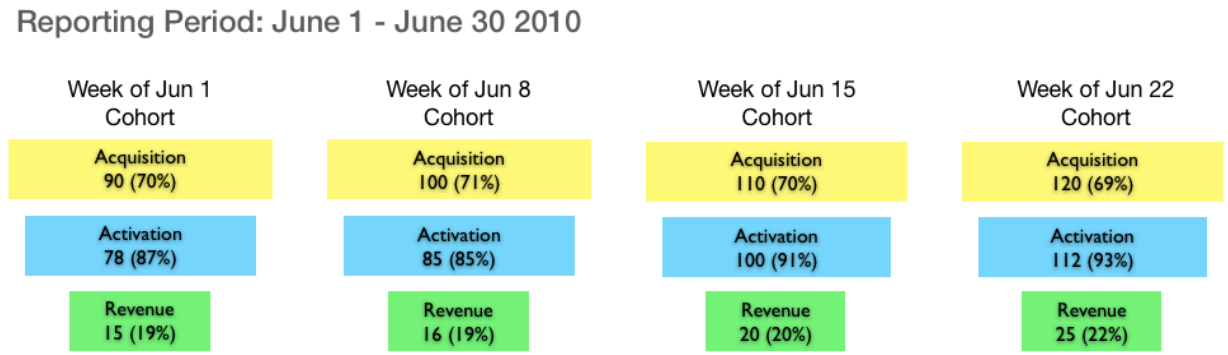
Lets see how cohort reports overcome the shortcomings with simple funnel reports:

The Weekly Cohort Report (by join date) shown below was generated using the same data used in the simple funnel report earlier (which I've shown again for comparison):

Simple Funnel Report:



Weekly Cohort (by join date):



You'll notice immediately that while the Acquisition and Activation conversion numbers are close enough, the Revenue conversion rates are very different.

Dealing with traffic fluctuations

Since all the events are tied back to the users that generated them, cohort reports handle fluctuations in traffic correctly.

Measuring progress (or not)

More importantly though, the weekly cohort report visibly highlights significant changes in the metrics which can then be tied back to specific activities done in a particular week.

Segmenting funnels

Since Cohort reports are inherently built around grouping users, they can be used to segment your funnels longitudinally around any property you track.

How to Build Your Conversion Dashboard

There are lots of third-party analytics products in the market. I have cut my teeth on Google Analytics, KISSmetrics, and mixpanel. Each tool has it's strengths and weaknesses, but unfortunately I haven't found a single analytics solution (yet) that address all the needs I outlined earlier. For these reasons, I tend to rely on a hybrid third-party + homegrown system for my conversion dashboard.

Rather than getting bogged down on the specifics of each tool, I'll present how I build my conversion dashboard from a functional perspective.

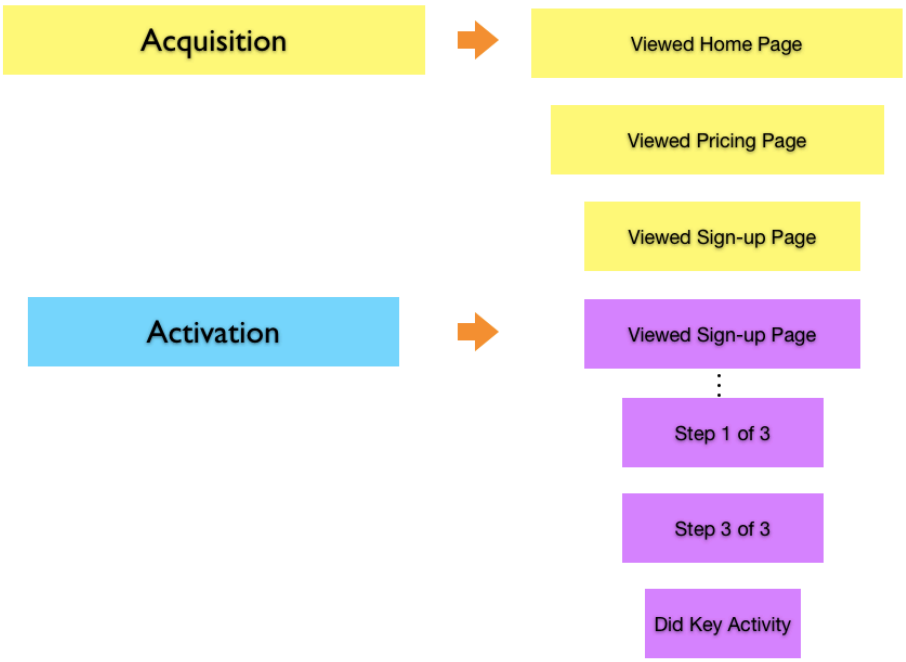
A key design principle is **decoupling data collection from data visualization**.

This lets you minimize waste by allowing you to build your conversion dashboard incrementally.

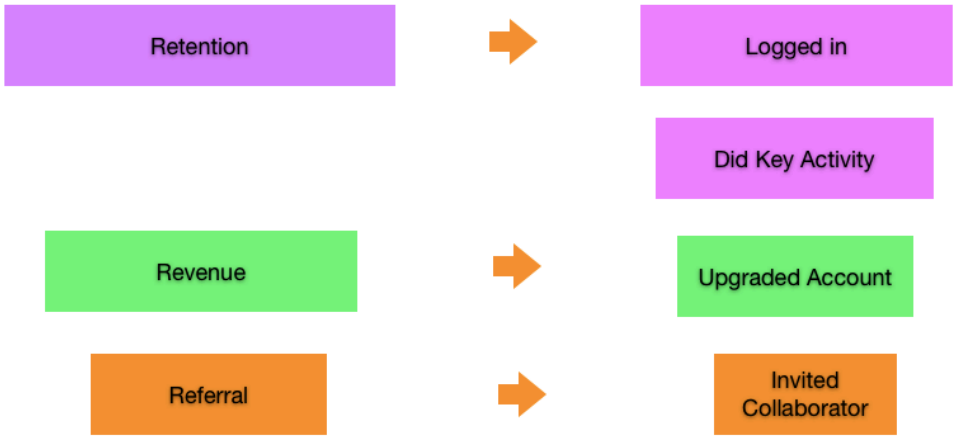
How to Collect Data

1. Map metrics to events

The first step is identifying all the key events (user actions) that map back to your metrics. You should already have all your steps for your Acquisition and Activation funnels clearly defined:



It is helpful to also identify any key events for the other macro metrics:



2. Track raw events

I recommend tracking your raw events in a separate events table/database or use a third-party system like Google Analytics, KISSmetrics, or mixpanel. While logging data in your production tables might seem easy and harmless enough, your production tables are probably not designed for the kinds of queries you'll need to run over time. You'll either end up spending a lot of time dumping tables and massaging the data, or taxing your production system heavily for reports.

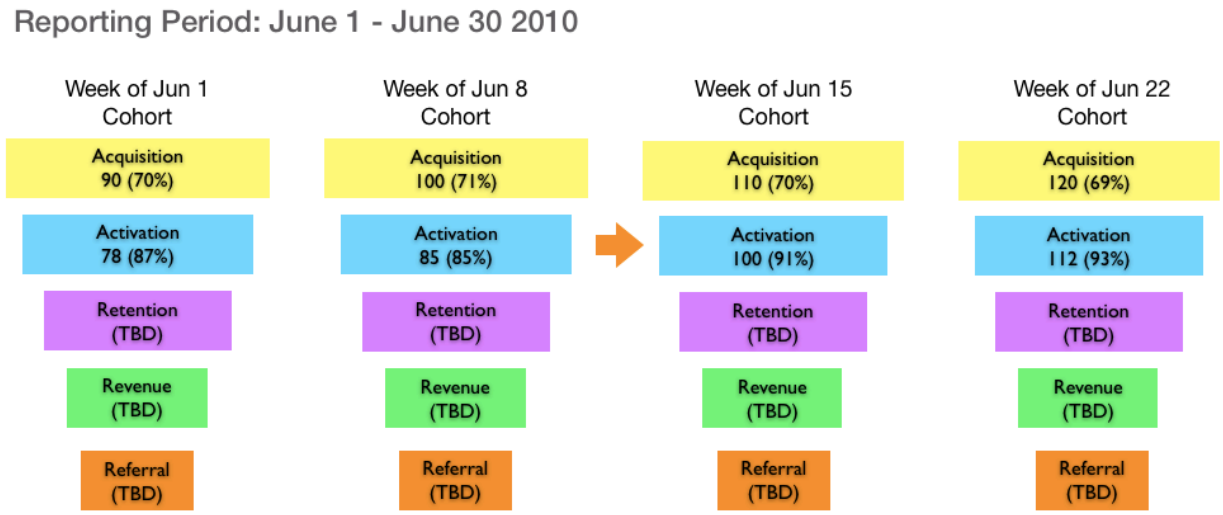
3. Log everything

A good practice to complement tracking raw events is logging every "potentially interesting" property along with each event. An example of a property could be your user's browser, operating system, referrer, etc. While you may not use a particular property today (or think you'll ever use it in the future), it's fairly inexpensive to log a few extra bytes of information that could end up saving you time later but more importantly provide a treasure trove of historical data.

How to Visualize Your Conversion Dashboard

1. Build a Weekly Cohort Report

The first report I use on my conversion dashboard is the weekly cohort report by join date I showed earlier:

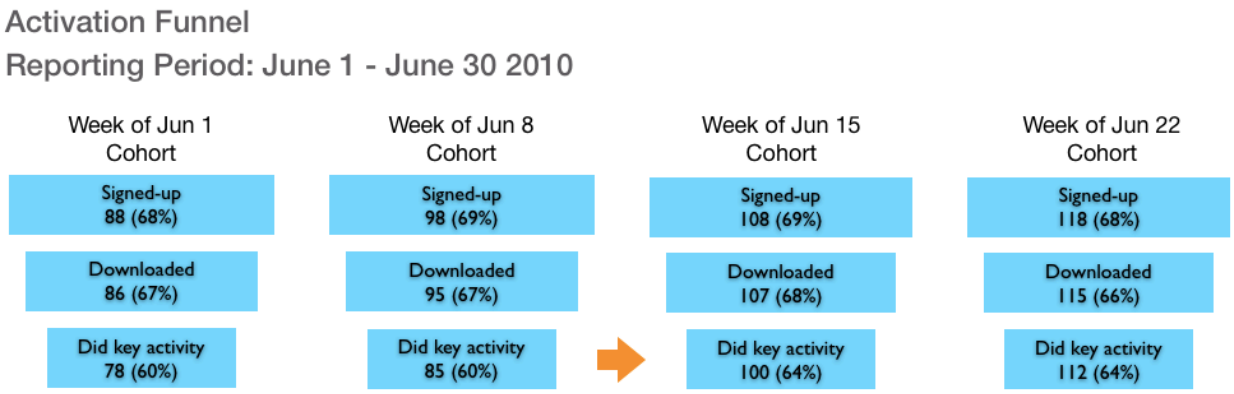


You'll notice that I base my Activation conversion rate off the number of "Acquired" users versus total visitors. This is because I like to measure my Activation rate (sign-up flow efficiency) independently from my Acquisition rate (marketing efficiency). This way if you get a surge of unanticipated PR traffic (like getting Digged or Techcrunched), unless these visitors truly have intent to use your service (i.e. click your Sign-up link), they will not affect your overall Activation numbers.

The Weekly Cohort Report serves like a canary in the coal mine. If you find none of your numbers changing from week to week, you are simply spinning your wheels and not really making progress. A change in the numbers in a particular week lets you tie back those results to actions taken in that week.

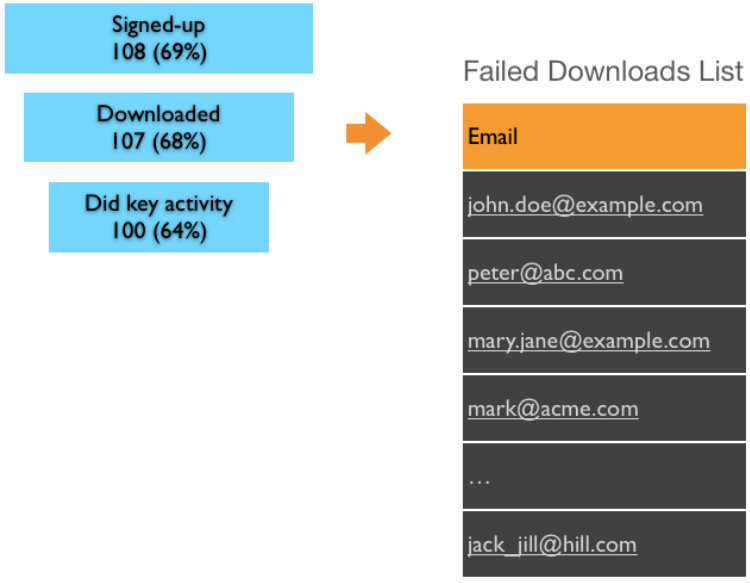
2. Be able to drill into your sub-funnels

You should be able to drill into your detailed sub-funnels and visualize all the steps which is valuable for troubleshooting problems:



3. Be able to go behind the numbers

At any given sub-funnel event, you should be able to go behind the numbers and get to the list of people:



CHAPTER 9

THE MVP INTERVIEW

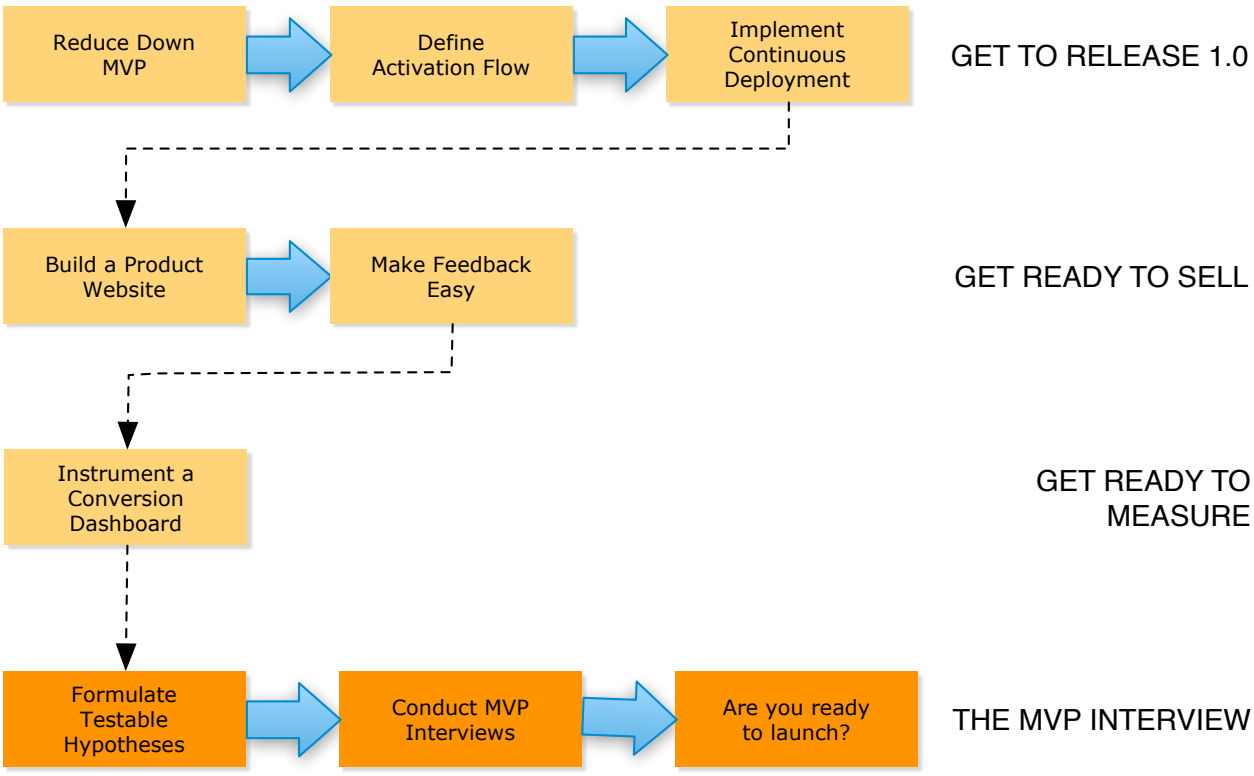
Before selling your MVP to strangers through your website, sell it to face-to-face to friendly early adopters. Learn from them. Then refine your design, positioning, and pricing for launch.

What You Need to Learn

Formulate Testable Hypotheses

Conduct MVP Interview

Are You Ready to Launch?


Reduce Down MVP
Define Activation Flow
Implement Continuous Deployment
GET TO RELEASE 1.0
Build a Product Website
Make Feedback Easy
GET READY TO SELL
Instrument a Conversion Dashboard
GET READY TO MEASURE
Formulate Testable Hypotheses
Conduct MVP Interviews
Are you ready to launch?
THE MVP INTERVIEW

What You Need to Learn

With your MVP, marketing website, and conversion dashboard ready, you are all set to pay your prospects another visit. Your objective is to sign them up to trial your service and in the process, test out your messaging, pricing, and activation flow.

If you can't convert a warm prospect in a 20 min face-to-face interview, it will be much harder to convert a visitor in under 8 seconds on your landing page.

During the MVP Interview, you are specifically looking to answer the following questions:

Unique Value Proposition: Compelling reason to buy

- *Does your landing page resonate with your early-adopters?*
- *Does it provide a compelling reason to buy?*

Revenue Streams: Pricing Model

- *Is the pricing model right?*

Solution: Activation Flow

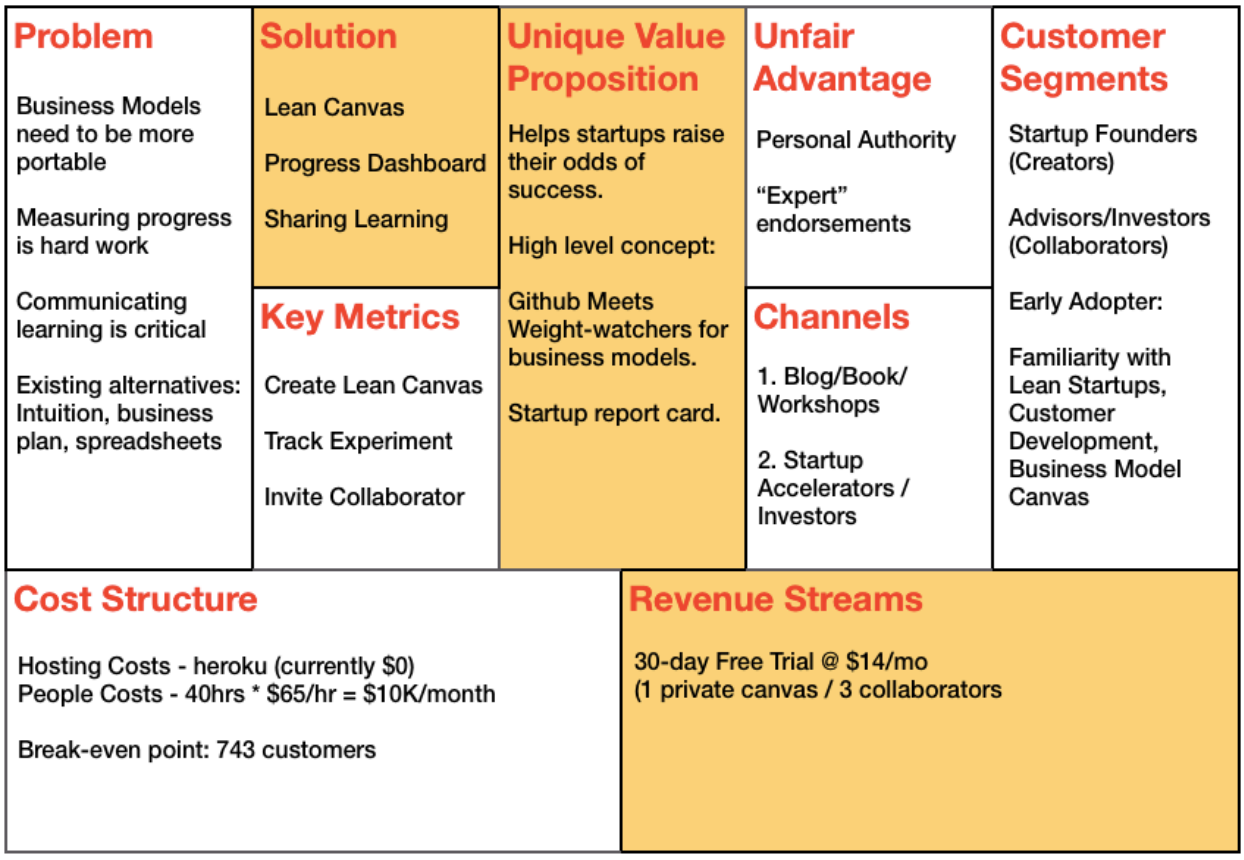
- *Do customers make it all the way through your Activation Flow?*
- *What are the usability hot spots?*
- *Does your MVP demonstrate and deliver on your UVP?*

Formulate Testable Hypotheses

By now you should come to expect this step.

Mini Case-Study: Lean Canvas

Here is our canvas from earlier with the sections under test highlighted:



Unique Value Proposition

Hypothesis: Early adopter will be able articulate UVP after 5 seconds test[3].
Hypothesis: Early Adopter will navigate to "Pricing" page (Acquisition sub-goal).

Revenue Streams

Hypothesis: Early adopter will agree to pay $14/mo for a 1 canvas/3 collaborator plan with a 30-day trial.

Solution

Hypothesis: Early adopters will be able to complete the Activation flow i.e. sign-up and successfully create a lean canvas (key activity).
Hypothesis: Early adopters will know what to do next.

---

[3] The 5 seconds test is a common usability test where you ask the subject to look at your landing page for 5-8 seconds, then look away and recount what they remember.

Conduct MVP Interviews

The MVP Interview, like the Problem and Solution Interviews, is less about pitching and more about learning. The structure of this interview largely follows a usability testing format described by Steve Krug in his book "Rocket Surgery Made Easy". I highly recommend getting a copy as you'll be conducting a lot more usability tests in part 3 of the book.

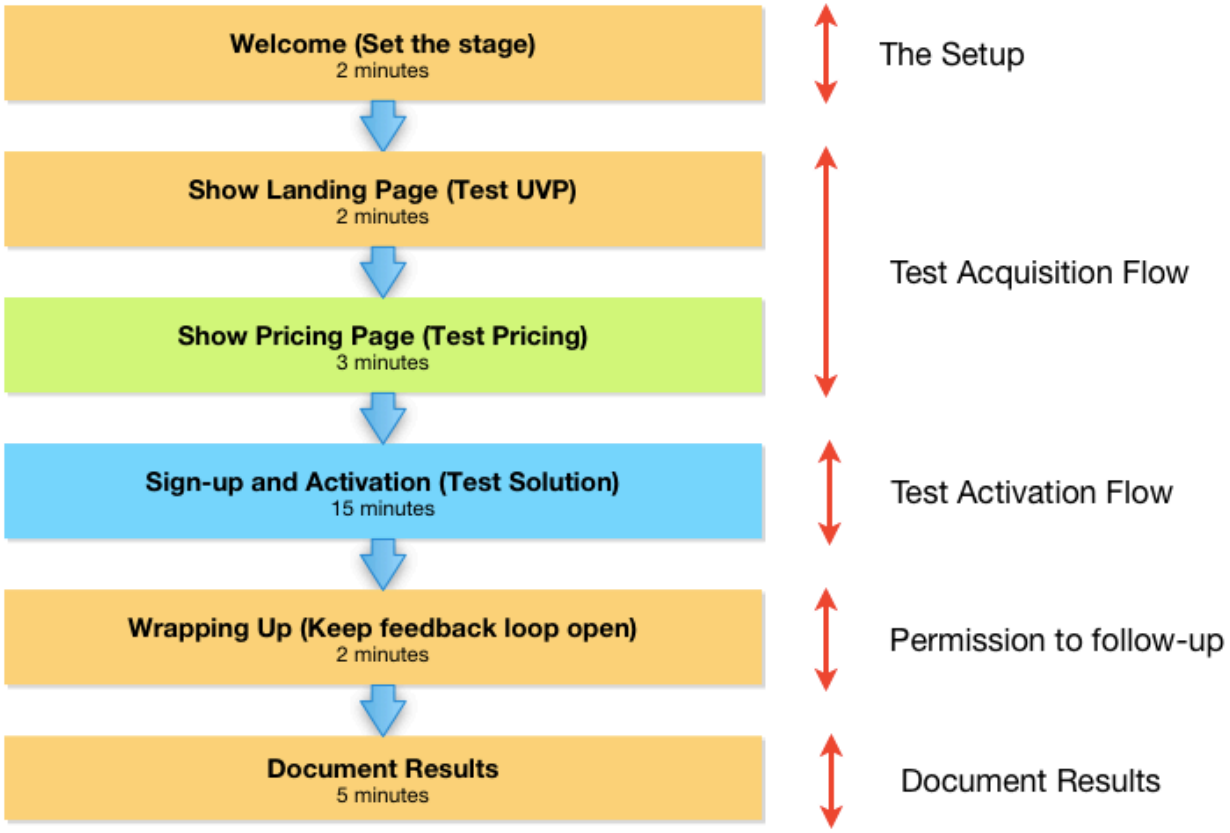
It is particularly important to conduct your initial MVP Interviews in person. Over time you might be able to do these with remote screen sharing software.

If your entire team cannot be present during the interview, I recommend using screen recording software (e.g. Camtasia, Screen Flow, etc.) to record the testing session for others to watch later.

Watching usability tests is like travel: it' a broadening experience.

- Steve Krug
Rocket Surgery Made Easy

The MVP Interview Outline



Welcome (Set the stage)
2 minutes

Briefly set the stage for how the interview works.

Thank you very much for taking the time to meet with us again.

We are almost ready to launch the business model validation tool we spoke about earlier. But before we launch, we wanted to show you the product, get your feedback, and if you're still interested give you early-access to the tool.

Does that sound good?

Great. We'd like to run the interview in a usability test format. So I'll start by showing you our website and ask you a few questions. It would be really helpful if you think out loud as we go along. That will help us identify any problems or issues we need to address.

Are you ready?

Show Landing Page (Test Unique Value Proposition)
2 minutes

Run 5 seconds test and test site navigation/call to action.

> Ok, we'll start with the home page. Please take a look at the home page and tell us what you make of it. Feel free to look around but don't click anything yet.
>
> Is is clear what the product is about?
>
> What would you do next?

Show Pricing Page (Test Pricing)
4 minutes

The interviewee should eventually end up on the pricing page where you can then ask them about your pricing model.

> Now, feel free to navigate anywhere on the site.
>
> <When they navigate on to the Pricing page>
>
> This is the pricing model we decided to launch with.
>
> What do you think of it?

Sign-up and First User Experience (Test Solution)
15 minutes

This is the heart of the interview.

Ask them to sign-up and watch how they navigate your Activation flow.

> Are you still interested in trying out this service?
>
> You can do so by clicking the “Sign-up” link.
>
> It would be immensely valuable to us if we could watch you go through the sign-up process. Would that be okay?

Wrapping Up (Keep the feedback loop open)
2 minutes

Hopefully the interviewee made it all the way through and you have a list of usability issues to address.

Congratulations, you have your first user!

Make sure they know what to do next and keep the conversation channel open with them.

> That's it. You're signed-up and ready to go.
>
> What did you think of the process?
> Is there anything we could improve?
> Do you know what to do next?
>
> Thank you very much for your time today. If you have any questions or run into any issues, please call us or drop us a note.
>
> Would it be okay if I check-in with you after you've had some time to use the tool some more. Say in a week?
>
> Great. Thanks again.

Document Results
5 minutes

As before, take the next 5 minutes immediately following the interview to document your results while they're still fresh in your mind.

Use the template on the following page to write down the top three problems you observed.

Have each person fill out this form independently and debrief later.

MVP INTERVIEW Date: .....

Contact Info

Name: .....

Email: .....

Usability Problem 1:

.....

.....

.....

Usability Problem 2:

.....

.....

.....

Usability Problem 3:

.....

.....

.....

Pricing

Willing to pay ($X/month): .....

Notes

.....

.....

.....

Are You Ready to Launch?

In this section I'll discuss how to make sense of your interview results, make any improvements, and determine when you are done.

1. Review your results every 5 interviews
Usability testing research shows that you can uncover 85% of your product's problems with as few as five testers.

2. Start with the most critical problems
Review everyone's top three problems and rank them by severity.

3. Do the smallest thing possible
Resist the temptation of completely redesigning a new landing page or sign-up flow at this stage. Your objective is to first establish a baseline that works and you can get there by making smaller tweaks. You'll have lots of opportunity to test alternate hypotheses in part 3 of this book.

4. Make sure things improve
Validate that your fixes actually improve things in subsequent interviews. Repeat Steps 1-3.

5. Audit your conversion dashboard
This is the perfect opportunity to audit your conversion dashboard and make sure everything works as expected.

What is the MVP Interview exit criteria?

You are done when your early adopters can consistently make it through your acquisition and activation flows.

Specifically, they should

- be able to clearly articulate your UVP,
- be primed to sign-up for your service,
- accept your pricing model,
- make it through your activation flow, and
- know what to do next.

3, 2, 1... Launch!

Once you have an MVP that works, your final step is revisiting your acquisition channel(s) to ensure you have a steady stream of prospects entering your funnel. However, be wary of spending a lot of effort prematurely optimizing your acquisition channels at this stage.

The "ideal" starting goal is driving 100 unique visitors to your landing page per day or 5-10 sign-ups a day. Strive to drive this traffic through the actual channels you've identified for your product (such as content marketing) but supplement with other means if needed (such as search engine marketing).

Your goal is establishing "just enough" traffic to support learning.

If you have large list of "warm" prospects from your earlier efforts (teaser page, referrals from interviewees), consider exhausting that list first in the form of more "early-access" signups before doing a public launch.

You don't need a lot of users to support learning.
Just a few good customers.

PART 3: PRODUCT/MARKET FIT

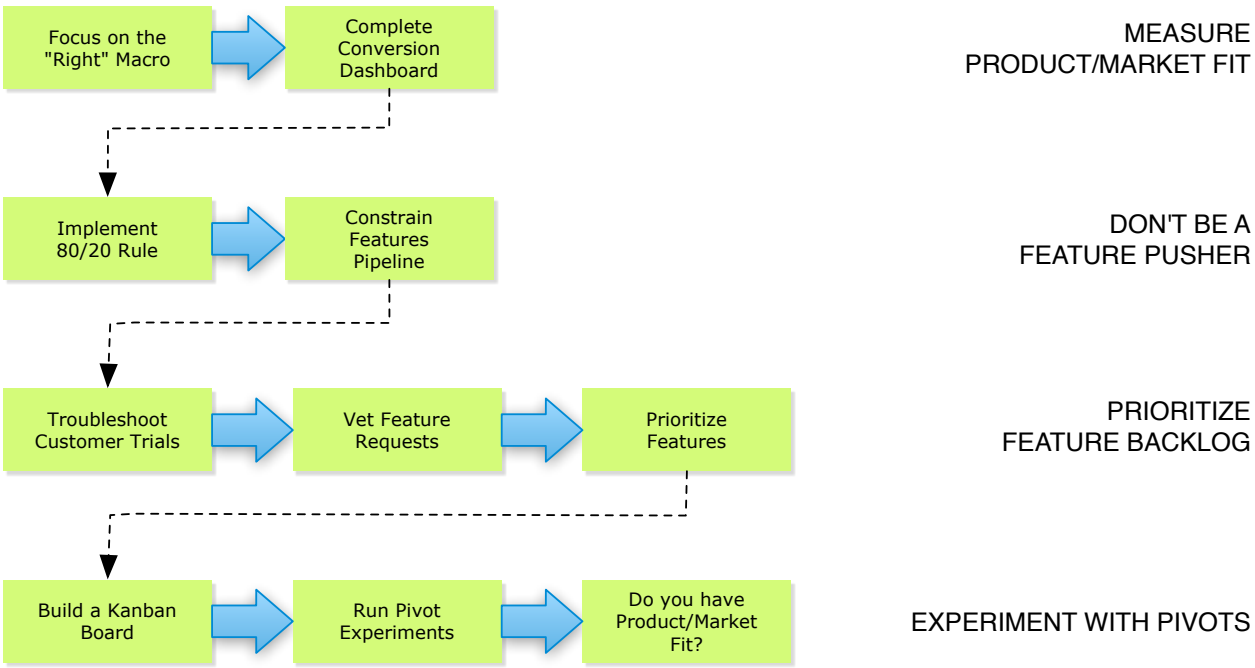
SUMMARY

Product/Market Fit, in one page.

Getting to Product/Market Fit is the first significant milestone for a startup.

The rubber hits the road after launch and this is when you truly start validating your complete product. While learning from customers is key, you have to know how to prioritize and then act on that learning.

The first step is defining a metric to measure Product/Market Fit. Once you have that, you can then systematically iterate towards achieving it.



CHAPTER 10

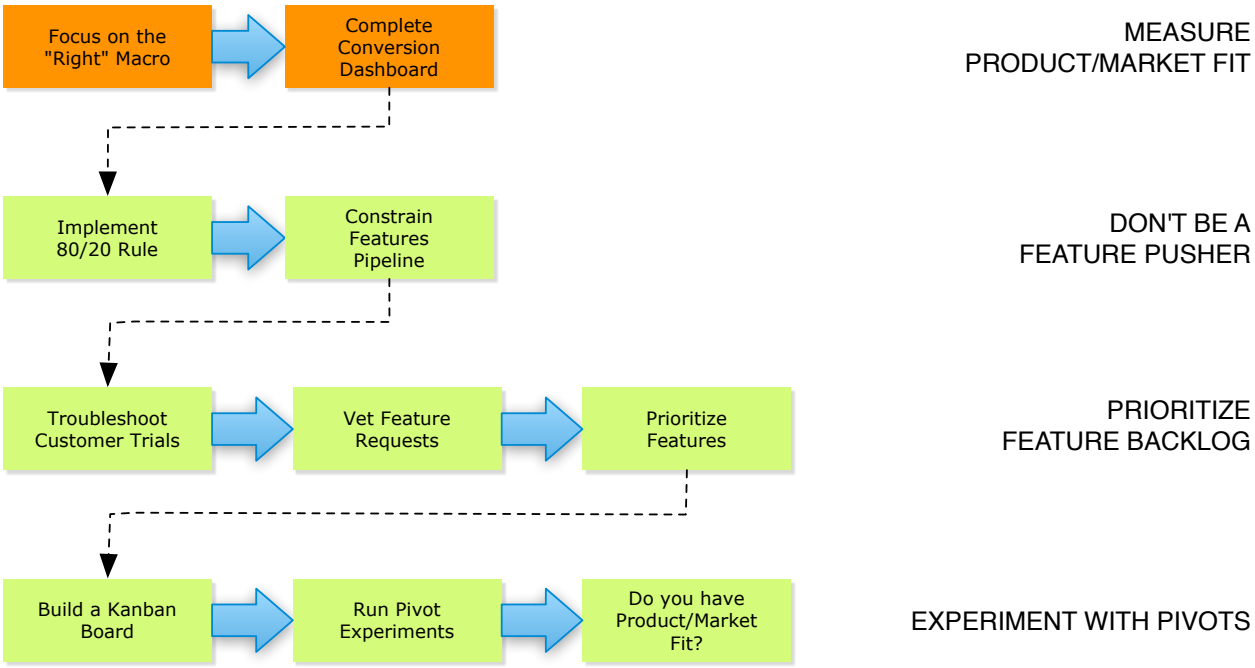
MEASURE PRODUCT/MARKET FIT

What is Product/Market Fit?

The Sean Ellis Test

Focus on the "Right" Macro

Complete Your Conversion Dashboard


Focus on the "Right" Macro
Complete Conversion Dashboard
MEASURE PRODUCT/MARKET FIT
Implement 80/20 Rule
Constrain Features Pipeline
DON'T BE A FEATURE PUSHER
Troubleshoot Customer Trials
Vet Feature Requests
Prioritize Features
PRIORITIZE FEATURE BACKLOG
Build a Kanban Board
Run Pivot Experiments
Do you have Product/Market Fit?
EXPERIMENT WITH PIVOTS

What is Product/Market Fit?

Even though Marc Andreessen may not have coined the term “ Product/ Market Fit”[4], his blog post on the topic remains one of the most popular descriptions of what Product/Market Fit feels like:

Product/Market fit means being in a good market with a product that can satisfy that market.

You can always feel when product/market fit isn't happening. The customers aren't quite getting value out of the product, word of mouth isn't spreading, usage isn't growing that fast, press reviews are kind of "blah", the sales cycle takes too long, and lots of deals never close.

And you can always feel product/market fit when it's happening. The customers are buying the product just as fast as you can make it -- or usage is growing just as fast as you can add more servers. Money from customers is piling up in your company checking account. You're hiring sales and customer support staff as fast as you can. Reporters are calling because they've heard about your hot new thing and they want to talk to you about it.

- Marc Andreessen
Pmarca Guide to Startups

Unfortunately, Marc ended that post with more questions than answers and didn’t offer any guidance on how to achieve or measure Product/Market Fit. Sean Ellis makes the concept less abstract by offering a metric for determining Product/Market Fit which I’ll cover next.

---

[4] The term “Product/Market Fit” was coined by Andy Rachleff who co-founded Benchmark Capital.

The Sean Ellis Test

Sean Ellis ran a consulting company, 12in6, that specialized in helping startups during their growth transition stage (post Product/Market Fit). As a condition to taking on a client, he conducted a qualitative survey across a sampling of the company's users to determine if their product had achieved Product/Market Fit.

The key question on the survey was:

> How would you feel if you could no longer use [product]?
>
> 1. Very disappointed
> 2. Somewhat disappointed
> 3. Not disappointed (it isn't really that useful)
> 4. N/A – I no longer use [product]
>
> If you find that over 40% of your users are saying that they would be "very disappointed" without your product, there is a great chance you can build sustainable, scalable customer acquisition growth on this "must have" product. This 40% benchmark was determined by comparing results across 100s startups. Those that were above 40% are generally able to sustainably scale the businesses; those significantly below 40% always seem to struggle.
>
> The Sean Ellis Test

I feel the exact wording of the question could use some slight tweaking depending on your target market. For example, in a B2B/Enterprise context, posturing to take away a product may may not sit well with your early customers who are investing time in your product. That aside, the

basic premise of the test is sound. It attempts to measure your product’s resonance with users.

The bigger challenge though with implementing Sean’s test is the same one I outlined earlier with customer surveys:

Surveys are more effective at verification than learning.

In this case, while Sean’s test can help **determine if** you have achieved Product/Market Fit, it doesn’t help you **achieve it**.

Additionally, for the results to be statistically significant, you need to have a large enough sample size, account for customer segmentation, and consider user motivation. For these reasons, the test is best administered closer to Product/Market Fit (which is also what Sean recommends).

So what do you do until then? How do you steer your product towards Product/Market Fit?

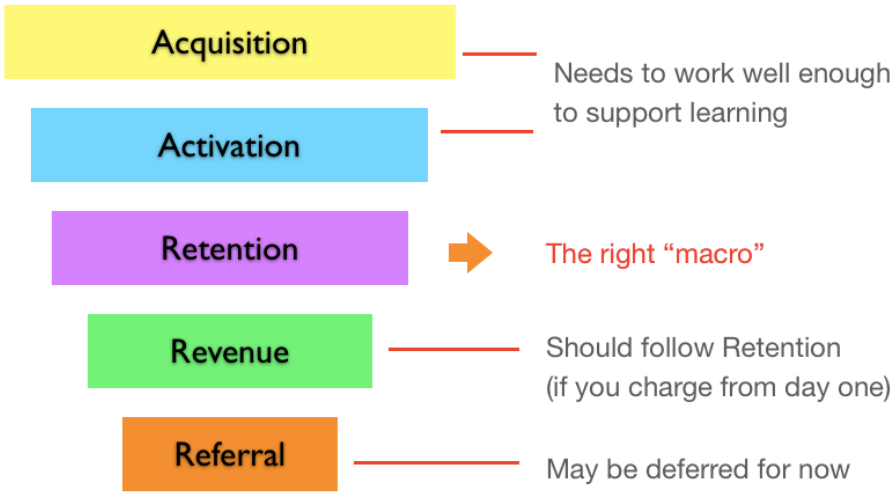
The answer lies within your conversion dashboard. In the next section, I’ll outline another approach to measuring your product’s resonance with users using a key metric from your User Lifecycle - **Retention**.

Focus on the "Right" Macro

"Build something people want."

- Paul Graham
Y Combinator

Achieving Product/Market Fit or traction can fundamentally be reduced to building something people want. Of the five startup metrics, the most indicative measure of "building something people want" is **Retention** i.e. repeated use of your product. Why else would they keep coming back?



The argument can also be made that repeated use of a product over a large enough period should correlate closely enough to Sean Ellis' "Very Disappointed" question. This makes it possible to apply the same 40% threshold to determining Product/Market Fit[5].

[5] I consulted Sean Ellis on this and he confirmed the correlation between 40% "Very Disappointed" and 40% Retention

You have achieved Product/Market Fit when you are retaining 40% of your activated users month over month.

The advantage of using a metric from your conversion dashboard to measure Product/Market Fit is that it gives you a macro goal and helps you **iterate** your product towards that goal.

Before you launch, measure what customers say.
After you launch, measure what customers do.

What about Revenue?

While I believe pricing is part of the product and advocate charging from day one, Revenue is only the first form of validation and could be a false positive as a Product/Market Fit test. I’ve experienced numerous cases with my products where customers kept paying for a product they did not use (not even sporadically). Sometimes this was because someone else was paying (their company, for instance) or that they simply forgot to cancel the product.

While Revenue is the first form of validation, Retention is the ultimate form of validation.

Furthermore, if you offer a subscription service and charge from day one **and** you have good Retention, Revenue will take care of itself.

Retention also holds up as the universal measure of **building something people want** even for products that delay or don’t charge their users.

What about the market in Product/Market Fit?

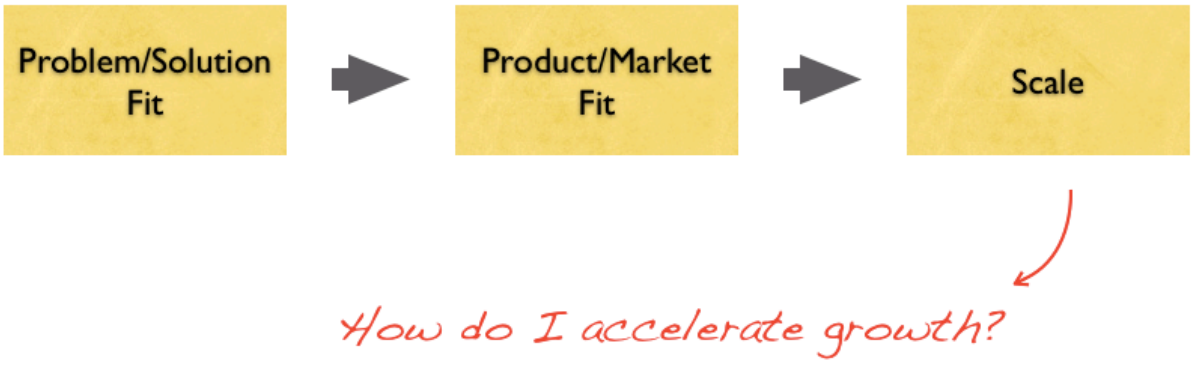
By that I mean things like market size, cost of customer acquisition, lifetime value - things that make a business model scalable.

Focusing on building a scalable business model at this stage is a form of waste.

During Problem/Solution Fit, you formulated hypotheses around your total market size but narrowed it down to first focus on your early adopters. You determined a price your early adopters would bear and factored in your cost structure to determine your break-even point.

Your first objective should be getting to strong early traction by
- retaining your early adopters,
- establishing an early beach-head,
- getting paid.

Only once you have achieved that, then shift your focus on scaling.



Complete Your Conversion Dashboard

In this section you’ll revisit your conversion dashboard and supplement it to track Retention. The good news is that you should already be logging all the raw event data needed to visualize it.

How to Track Retention

Retention measures repeated activity over a time period. So the first step is defining what constitutes activity.

1. Define an Active user
There are many ways to define an Active user. The most basic definition measures activity simply in terms of logins i.e. does the user come back?

A more representative definition for tracking activity for Product/Market Fit should not just measure usage but “representative usage”. This is where the “Key Activity” block on your Lean Canvas comes into play. Every product has a core set of user actions which track ongoing “representative usage”. For example, writing blog posts is a key activity for a blogging platform. Also note that your key activity for Activation may not be the same as your key activity for Retention.

A more advanced approach to measuring “representative engagement” comes from Dharmesh Shah who coined the term Customer Happiness Index or CHI. The idea is to use a formula to grade activity on a scale of 1-100 that is calculated using frequency, breadth, and depth of feature usage.

At this stage, I recommend starting with the simplest formula that measures “representative engagement” which centers around your key activity.

You can tweak the formula for your product over time to get a more graded CHI score that helps you segment your users into different buckets and focus your marketing, troubleshooting, and customer development activities.

Case Study: Lean Canvas

The key activity in Lean Canvas that tracks ongoing usage is the creation of experiments.

I would start by simply defining an active user as someone who creates at least one experiment during the trial period (30 days).

A more advanced approach might be calculating a Customer Happiness Index using a weighted formula like the one below:

CHI = [(Number of days logged in)/(Desired number of logins)*0.2 +
(At least one key activity) *0.8)] * 100

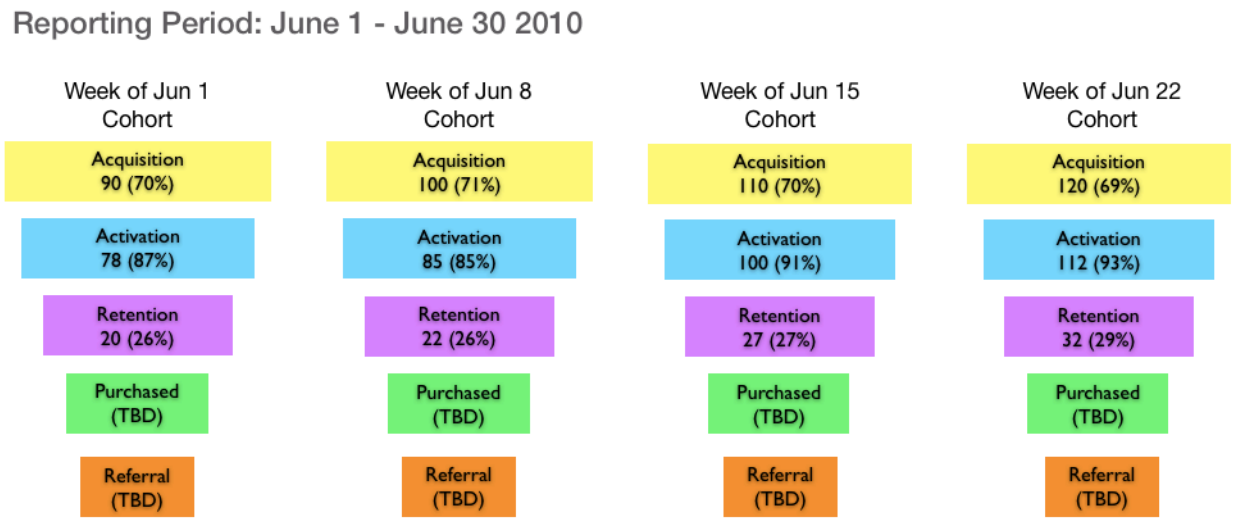
And then defining an Active user as someone with a CHI > 80. While this yields the same number of Active users as before, it gives me a graded scale for segmenting my users by activity.

Here is what four users with varying levels of activity over the trial period would look like:

User	Number Logins	Key Activity	CHI	Active
A	20	2	93	YES
B	10	1	87	YES
C	5	0	3	NO
D	15	0	10	NO

2. Visualize Retention in your Conversion Dashboard

Now that you have a definition of an Active User you can use visualize your conversion dashboard to show what percentage of users were active during your trial period.



The Retention rate is based off the number of “Activated” users.

3. Provide a detailed view

As with your other macro metrics, drilling into the Retention macro should provide a detailed view. However, in this case instead of showing a sub-funnel, you would show the trending of your retention numbers over time.

Week joined	1 month later	2 months later	3 months later	4 months later	5 months later	6 months later	7 months later
Jun 1	26%	24%	22%	20%	20%	20%	20%
Jun 8	26%	25%	24%	22%	22%	22%	?
Jun 15	27%	26%	25%	23%	23%	?	
Jun 22	29%	27%	27%	25%	?		
Jun 29	32%	30%	30%	?			
…	…	…	…	…	…	…	…

Note: You should ideally be able to change the time periods on both axes - allowing you to visualize this report by day, week, or month.

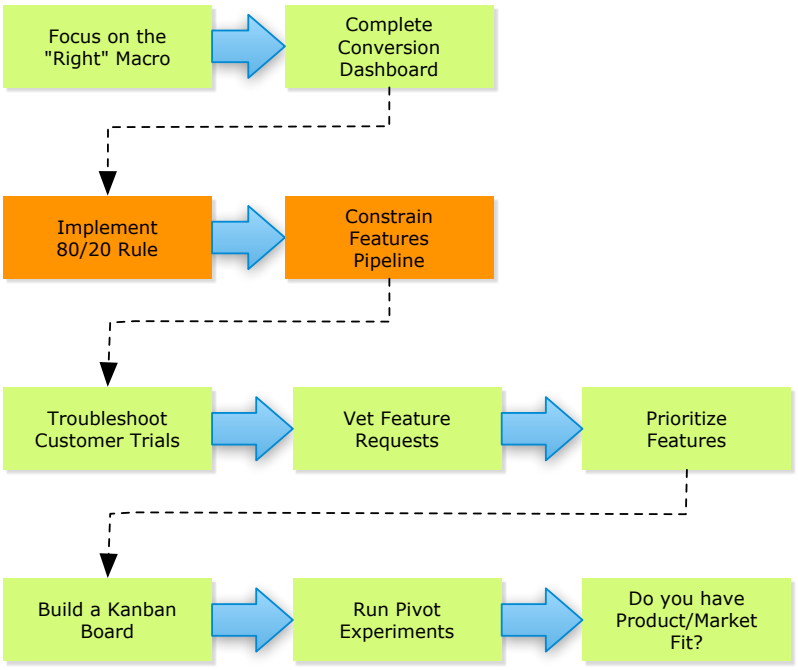
CHAPTER 11

DON'T BE A FEATURE PUSHER

Features Must Be Pulled, Not Pushed

Implement 80/20 Rule

Constrain Your Features Pipeline



MEASURE PRODUCT/MARKET FIT

DON'T BE A FEATURE PUSHER

PRIORITIZE FEATURE BACKLOG

EXPERIMENT WITH PIVOTS

Features Must Be Pulled, Not Pushed

In a great market -- a market with lots of real potential customers -- the market pulls the product out of the startup.

- Marc Andreessen
Pmarca Guide to Startups

During Product/Launch Fit, I advocated implementing a Continuous Deployment system. While Continuous Deployment helps you streamline your product development process for speed, you have to be wary of simply cranking out more features faster.

When you launch your product, lots of things can and will go wrong. Sure enough, feature requests will also start pouring in. The common tendency is to build more, but that is seldom the answer.

Here's why:

More features dilute your Unique Value Proposition
You have taken great effort to keep your MVP as small as possible. Don't dilute your MVP with unnecessary distractions.

Simple products are simple to understand.

Don't give up on your MVP too early
Building great software is hard. While you have painstakingly tested problems worth solving, you have only tested a semblance of the solution. Give your MVP a chance. First troubleshoot and resolve issues with existing features before chasing new features.

Put down the compiler until you learn why they're not buying.

- Jason Cohen
A Smart Bear

Features always have hidden costs

More features mean more tests, more screenshots, more videos, more coordination, more complexity, more distractions.

Start With No.

- 37signals
Getting Real

You still don't know what customers really want

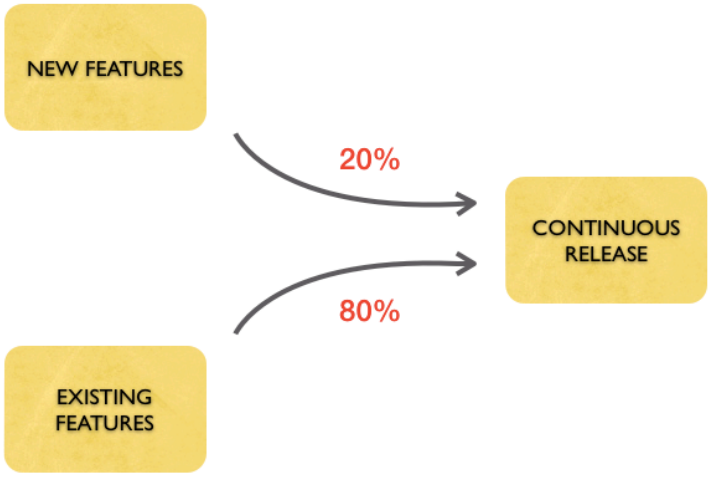
Treat your future feature ideas like experiments. Keep them on your feature backlog for now. I'll cover how you prioritize, build, and validate new features shortly.

Feature creep can become an addiction.

- Ben Yoskovitz
instigator blog

Implement 80/20 Rule

A good rule of thumb for prioritizing focus is implementing a 80/20 Rule:



Most of your time immediately after launch should be spent measuring and improving existing features versus chasing after new shiny features.

But even with this breakdown, it's possible to keep cranking out improvements that have zero impact.

The next section helps with that.

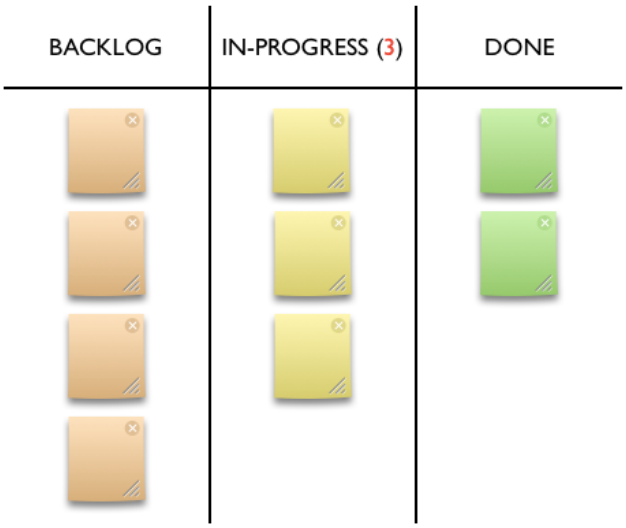
Constrain Your Features Pipeline

A good practice for keeping your features pipeline in check is limiting the number of features that can be concurrently worked **and** only working on new features after you've validated that the features you just deployed had a positive or negative impact i.e. yielded learning.

A great way for doing this is with a Kanban[6] board (or visual board).

A Kanban board is to feature tracking as a Conversion Dashboard is to metrics tracking. Both let you focus on the Macro.

A very basic Kanban board is shown below with 3 buckets:



[6] Kanban is a scheduling system designed by Taiichi Ohno, father of the Toyota Production System, that tells you what to produce, when to product, and how much to product. (Source: Wikipedia)

The basic idea is that features start on the left-hand side and move through stages of product and customer development before they are considered "Done".

Here is a high-level overview of the three basic process steps shown here:

1. Backlog
All potential features start life in the Backlog bucket. They get in there in one of the following ways:

- Existing feature improvements e.g. refined sign-up flow
- Customer feature requests
- Your feature requests e.g. the nice-to-have's you deferred earlier

Before going further, it is important to make a distinction between Minimal Marketable Features (MMFs) and smaller features/bug fixes. MMF was first defined in the book "Software by Numbers" as the **smallest portion of work that provides value to customers.**

By feature, I always mean a Minimal Marketable Feature (MMF). A good test for a MMF is to ask yourself if you'd announce it to your customers in a blog post or newsletter. If it's too tiny to mention, then it's not a MMF.

A MMF is typically made up of smaller work items (tasks) which, if you are implementing continuous deployment, define your small batches. Smaller features and bug fixes typically fit within a work item or small batch.

I only track MMFs on a Kanban board and use a lighter-weight task board tool (like Pivotal Tracker) to track smaller features, bug fixes, and work items.

2. In-Progress

The Backlog queue is usually kept in priority order based on the current goals (focus) of your product. This makes it easy to simply pick the top feature in the list and being work. The "In-Progress" step is in turn made up of several sub-steps such as building mockups, coding, deploying, etc. I'll cover these details in a later chapter.

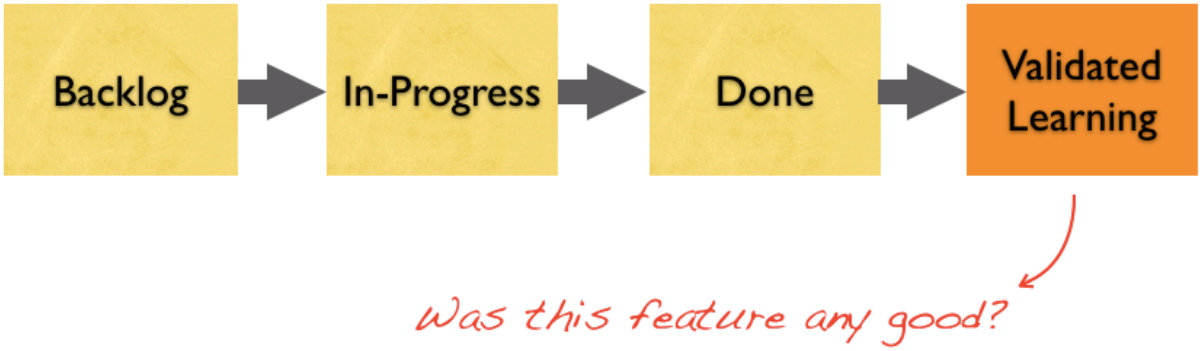
A key principle of Kanban that works to constrain the work queue is setting limits on the number of features that can be in progress at any given time. This allows you to maximize on throughput while minimizing on waste. For the technically inclined, Donald Reinersten's book: "The Principles of Product Development Flow" covers why this is so in great detail.

I recommend starting with a work-in-progress limit equal to the number of founders/team members and adjusting later if you need it. So if you have 3 founders, only 3 features can be worked on at any given time.

3. Done

When the feature is done, it's moved into the "Done" bucket. The "Done" state is somewhat arbitrary and different software development teams use "Done" to mean anything from "Code Complete", to "Tested", to "Deployed".

In a Lean Startup, however, a feature is only "Done" when it **provides validated learning from customers**:



For this reason, Eric Ries suggests either defining "Done" to include validated learning or adding a fourth state for validated learning. As we'll see a little later, I do a bit of both using a two phase validation - first qualitatively, then quantitatively.

Defining "Done" this way further constrains your feature pipeline and prevents you from working on any new features unless you can prove that the current features just deployed provided validated learning.

Some additional thoughts

In a typical Agile Software Development process, you time-box iterations (typically two weeks), define and estimate stories to fit within that iteration. After the iteration you cut a release. There is a push to make the iteration as small as possible which necessitates that stories also be kept small. This in turn explodes the number of stories and inevitably leads to story thrashing and spill-over across iteration boundaries.

Having implemented a 2 week release cycle for a number of years and then switched to Continuous Deployment, I find it unnecessary to take on the added overhead of managing time-boxed iterations anymore.

Kanban coupled with Continuous Deployment allows you to track a feature as it's own iteration (or experiment). This results in fewer stories, better tracking of features, and less overhead.

Progress that used to be measured in terms of "team velocity" is now replaced by "cycle time" which I find simpler to comprehend and more accurate for planning.

In the next chapter I'll cover how you build your feature backlog and in the chapter following I'll cover the full feature lifecycle in detail.

CHAPTER 12

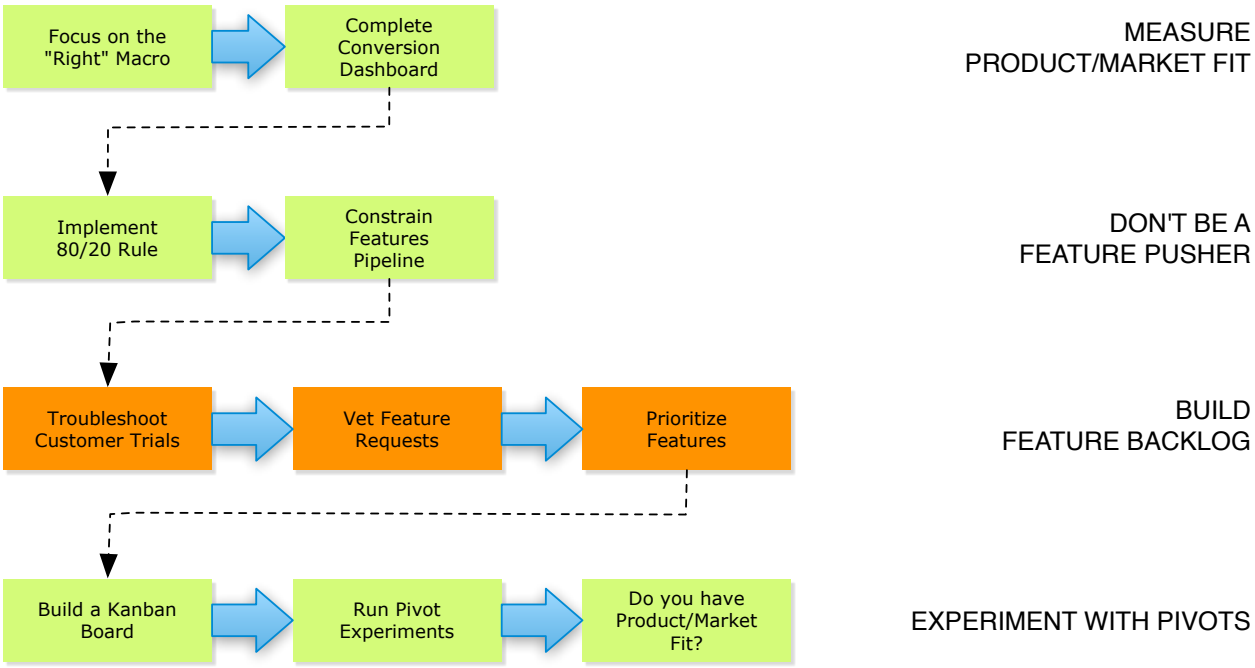
BUILD YOUR FEATURE BACKLOG

The trial period is a critical window of time when prospects grant you their attention and permission to engage them (within limits). Trials time-box the full user lifecycle and force an outcome which leads to quick actionable learning. Properly done trials are a goldmine opportunity for learning, but they can just as easily be fumbled.

Troubleshooting your trials is also the best way to build up your feature backlog.

Troubleshoot Customer Trials

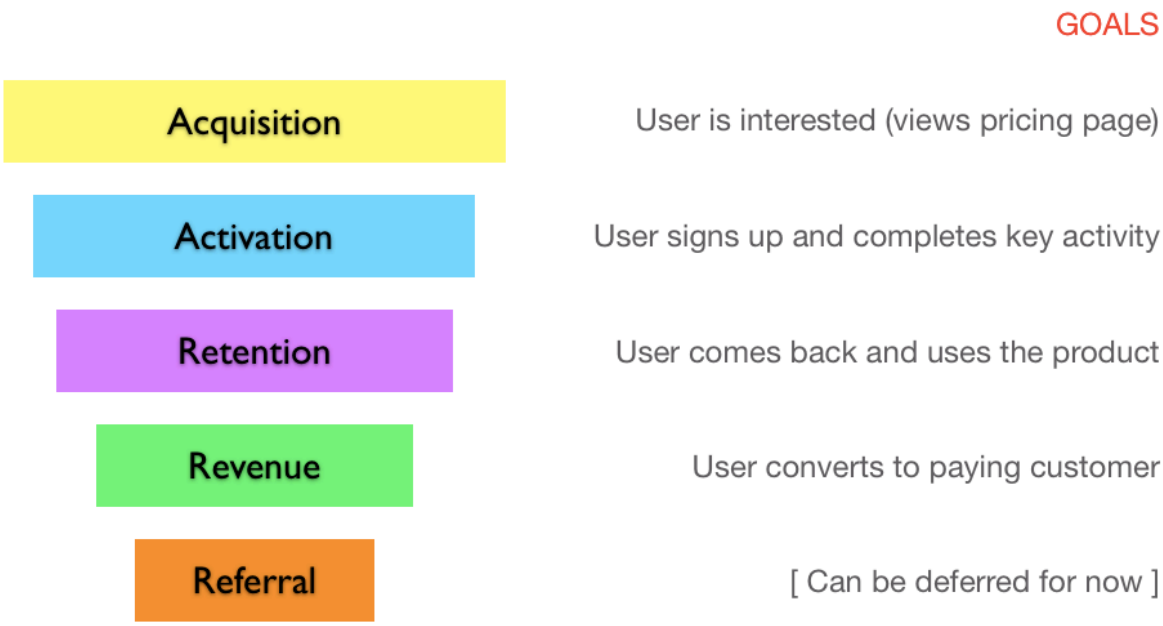
Process Feature Requests


Focus on the "Right" Macro
Complete Conversion Dashboard
MEASURE PRODUCT/MARKET FIT
Implement 80/20 Rule
Constrain Features Pipeline
DON'T BE A FEATURE PUSHER
Troubleshoot Customer Trials
Vet Feature Requests
Prioritize Features
BUILD FEATURE BACKLOG
Build a Kanban Board
Run Pivot Experiments
Do you have Product/Market Fit?
EXPERIMENT WITH PIVOTS

Troubleshoot Customer Trials

I particularly like trials because they time-box the full user lifecycle and force an outcome which leads to quick actionable learning. The way you troubleshoot your trials is following the path a user takes through your user lifecycle.

Your first objective during trials is reducing user abandonment on your Acquisition and Activation paths. Your next objective is increasing retention and engagement. And finally getting paid.



Acquisition and Activation

Priority: Ensure you are driving enough traffic to support learning.

You qualitatively tested your Acquisition and Activation flows during Product/Launch Fit and the hope is that they still work well enough to drive sufficient traffic for learning. But since you might now be driving new unaware visitors to your landing page, you need to revisit these metrics again.

1. Drill into your sub-funnels

Explore your Acquisition and Activation sub-funnels to see where users are dropping off.

a. Start with the leakiest bucket first. Are you loosing them on a particular page e.g. landing page, pricing page?
b. Also analyze the properties you collect with each event and look for patterns. Do certain types of users (for example, mac versus windows users) experience higher failure rates than others?

2. Reach out to your users

You should be able to extract the list of users that failed at a particular step in your funnel. If you know what went wrong, correct it, and ask those users to come back. If you don't know what went wrong, reach out with an offer for help (more like call for help).

3. Catch and report unexpected errors

When early users run into problems, they don't turn into testers. They leave. To be able to still learn from their experience, catch and report unexpected errors so you can troubleshoot the problem without them.

Retention

Priority: Get users to come back and use your product during trial

The 40% Retention rate for Product/Market Fit needs to be measured over a longer time window than your trial. Your immediate goal is to get activated users back to use your product at least a few times before the trial window expires. A good initial goal is to aim for a 25% Retention rate.

1. Send gentle email reminders

Email is a very effective (and often under utilized) medium for engaging your customers. Everyone has an email address. Email can be automated, tracked, and measured.

A common technique used by email marketers is “drip marketing” where you schedule a set of pre-determined messages to your users over time. Even interested users get busy and distracted and gentle reminders can help bring them back to your product.

But even better than “drip marketing” is “lifecycle marketing”. Lifecycle marketing additionally takes the user’s stage in the User Lifecycle into consideration. So for instance, if a user gets stuck during activation, instead of educating them about your advanced features, you would know to send them timely and appropriate troubleshooting help.

2. Follow up with your interviewees

During the MVP Interview, you asked for permission to follow up with your early adopters. Follow through. Call them up or meet with them and get their feedback.

Revenue

Priority: Get Paid

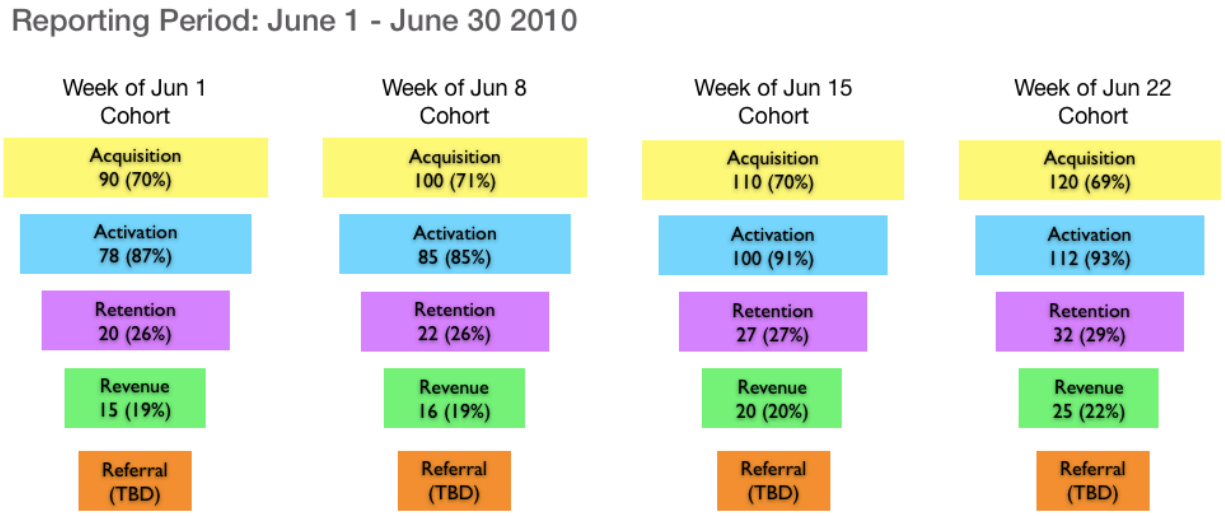
Getting paid is the first form of validation of your product and the end of the trial forces an outcome one way or the other. This is where the rubber hits the road and you stand to **learn from either outcome**.

1. Implement payment system

Now is the time to implement a payment system for customers to pay you.

2. Track Revenue on your Conversion Dashboard

Once you start getting paid, visualize your customers' first revenue event on your Conversion Dashboard.



3. Track Revenue over time in a detailed view

Much like retention, you should also track and trend your revenue over time. This report will be valuable in extracting things like your lifetime value of customers, average time to conversion, etc.

Week joined	1 month later	2 months later	3 months later	4 months later	5 months later	6 months later	7 months later
Jun 1	$735	$665	$630	$613	$613	$613	$613
Jun 8	$784	$756	$714	$700	$700	$700	?
Jun 15	$980	$945	$910	$875	$875	?	
Jun 22	$1,225	$1,190	$1,138	$1,120	?		
Jun 29	$1,470	$1,435	$1,400	?			
…	…	…	…	…	…	…	…

Note: You should ideally be able to change the time periods on both axes - allowing you to visualize this report by day, week, or month. I also find it helpful to switch between actual revenue and percentage conversions.

3. Get paying customers to talk to you

Get them on the phone, thank them for upgrading, and ask them:

- how they heard about you? (if you don't know)
- why they bought?
- what could be improved?

4. Get "lost sales" prospects to talk to you

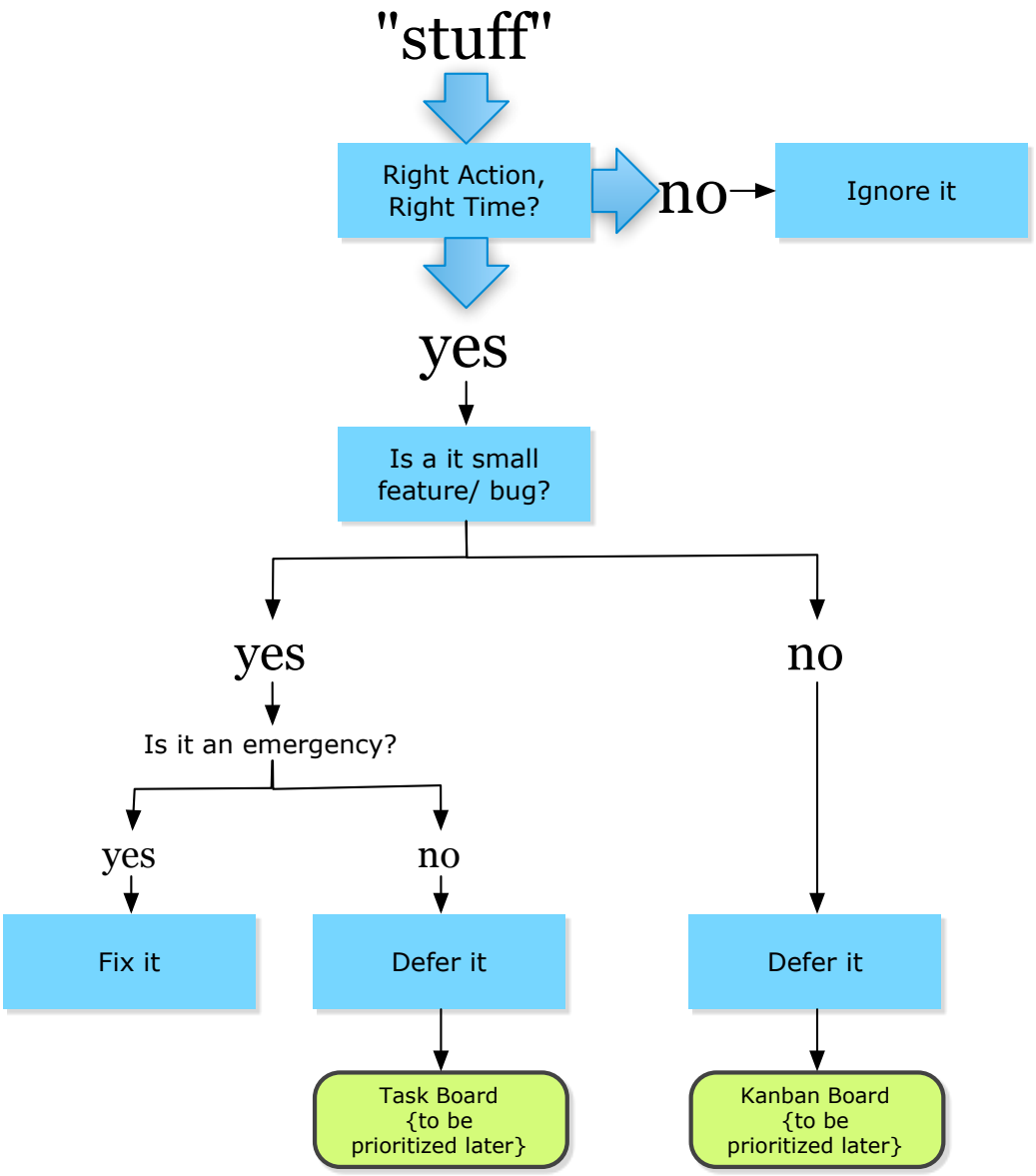
You stand to learn as much (if not more) from your lost sales. While some are happy to provide honest feedback if you make a sincere request at the end of the trial, others might need to be slightly incentivized. Offer a $25-50 gift-card or donation to charity in exchange for 15 minutes of their time.

Don't spend a lot of effort acquiring customers and then just let them walk away.

- Gary Vaynerchuck

Process Feature Requests

In this section, I'll outline a GTD style workflow for how to process new work requests that will inevitably come up during the trial period.



The first determination is checking the request against your product's immediate needs and priorities - Is it Right Action, Right Time? So for instance, if you have serious problems with your sign-up flow, all other downstream requests should take a backseat to that.

After that you need to consider whether this is a small feature/bug fix or a larger Minimal Marketable Feature (MMF).

If this is small work item and something that is needed immediately, fix it right away i.e. code-test-deploy using your continuous deployment process. Otherwise, add it to your task board's backlog bucket. I recommend also keeping the task board backlog in priority order. That way anyone on the team can simply pull off a small work item and push it all the way through deployment when they have some idle time.

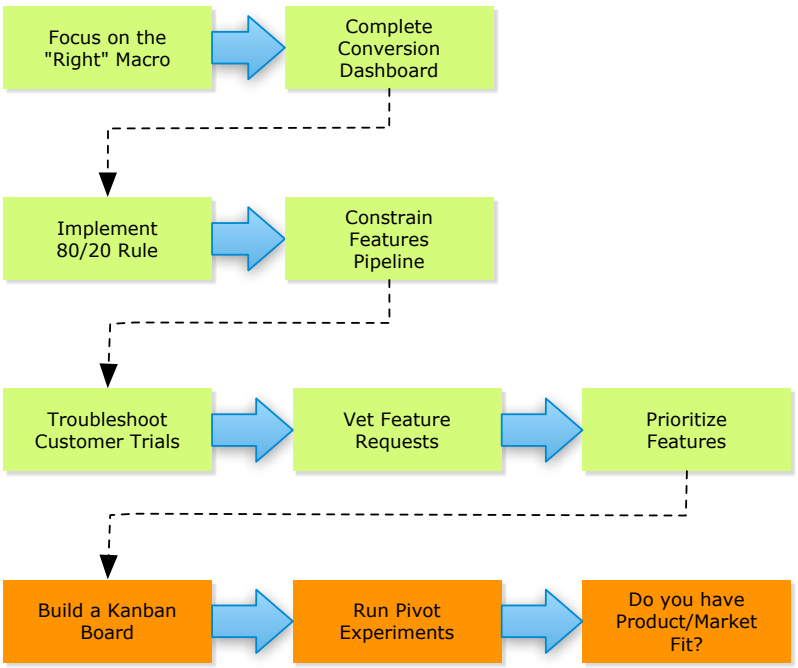
If this is a larger MMF, it goes on your Kanban board's backlog bucket.
In the next section, I'll cover how you prioritize and work on these features.

CHAPTER 13

EXPERIMENT WITH PIVOTS

The Feature Lifecycle

Have You Built Something People Want?



MEASURE
PRODUCT/MARKET FIT

DON'T BE A
FEATURE PUSHER

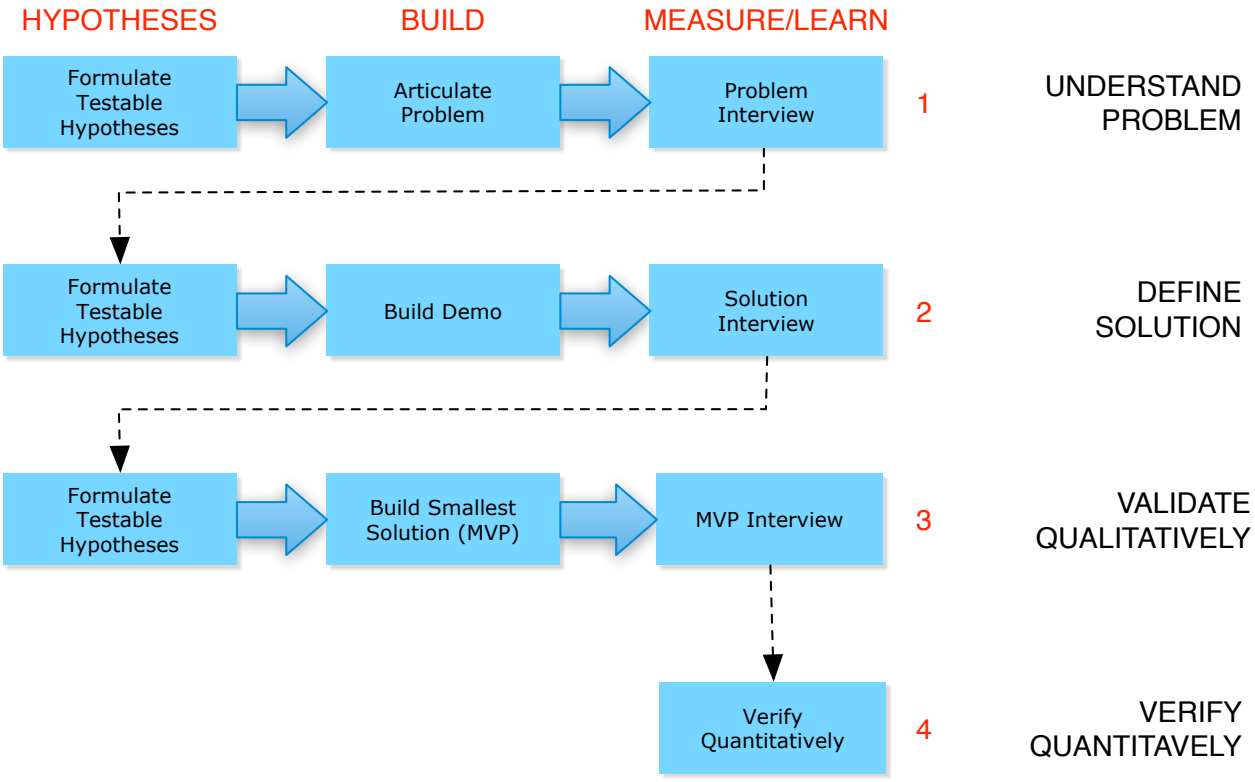
BUILD
FEATURE BACKLOG

EXPERIMENT WITH PIVOTS

The Feature Lifecycle

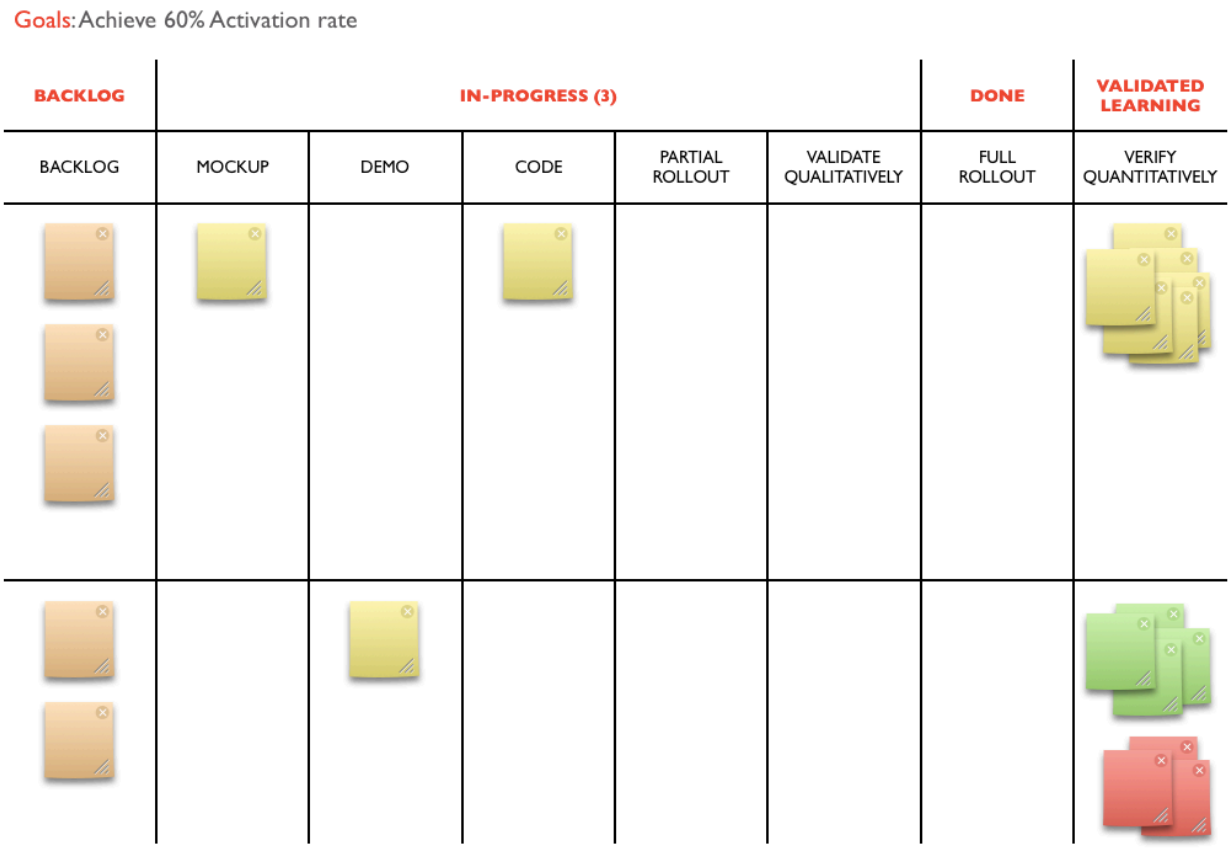
The iteration meta-pattern we have been using to define, build, and validate your MVP also applies to MMFs (Minimally Marketable Features).

In this section, I'll outline a feature lifecycle built on this meta-pattern and implemented using a Kanban board:



How to Track Features on a Kanban Board

Before I get into the specifics of the process steps, I'd like to highlight some general aspects of the Kanban board first.



Goals

It is a good idea to list your immediate goals and priorities (focus) at the top of your Kanban board. This helps keep everyone on the same page when prioritizing your backlog.

Work-In-Progress limits

The work-in-progress limit is shown in the top header row. It is typical in larger teams to also set limits on each sub-state (mockup, demo, code, etc.) but that is overkill at this stage since most startup teams are small.

Buffer lanes
Each process step is divided into two parts. The top section is used for features currently "under work", while the bottom section (also called the buffer) is used to hold features that have been "completed" and waiting to be picked up for the next process step.

Features can be killed at any stage
There are multiple customer validation stages built into the feature lifecycle. If a feature fails validation, it can either be moved back to the previous stage to be reworked, or killed. Features slated to be killed are marked in red.

Continuous Deployment
I assume you are following a continuous deployment process and group the COMMIT-TEST-DEPLOY-MONITOR cycle simply under CODE.

Two Phase Validation
Because quantitative verification can take a while, I only use qualitative testing to declare a feature as "Done". This releases the work-in-process lock on that feature so other features can be worked while more data is collected.

The Process Steps Explained

Now I'll describe the full feature lifecycle through the process steps:

UNDERSTAND PROBLEM

1. Backlog

We finished the last chapter with a simple workflow for quickly vetting feature requests for your backlog. These are placed in the top part of the Backlog column since they aren't started yet. Because you have a finite work-in-process limit, you need to carefully prioritize your backlog queue against your product's immediate goals.

Once you have identified a feature, the first step is testing to see if the **problem is worth solving**. If you can't justify building the feature, kill it immediately.

a) Customer Pulled Requests

If the feature is a customer pulled feature, arrange a call or meeting with the customer. Even though the customer might be asking for a specific solution, get to the root problem. Try and talk them out of the feature. Have the customer sell you on why you should add the feature.

At the end of the call, you should be able to assess if this is a nice-to-have or must-have problem, whether it is worth solving, and which macro it will affect.

b) Internal Requests

If the feature was internally generated, review the same criteria as above with other team members and similarly get to a **"Is this worth solving?"** determination for this feature.

DEFINE SOLUTION

2. Mockup

Once you have a feature worth building, build a mockup first using the same approach outlined in the Solution Interview chapter. Start with paper sketches but quickly get to HTML/CSS views that are ideally accessible from within your application.

3. Demo

With the mockup ready, conduct an interview similar in structure to the Solution Interview that tests your solution with customers. Iterate as needed on the mockup till you have a strong signal to move forward.

4. Code

With the mockup validated, you can now start building the functionality behind the feature. It will most likely make sense to break the feature into a number of smaller work items which you can track using your Task board and deploy incrementally using your Continuous Deployment system.

VALIDATE QUALITATIVELY

5. Partial Rollout

Once the feature is coded and ready for use, partially deploy it to just a few customers first.

6. Validate Qualitatively

Conduct usability interviews similar to the MVP Interview. Iterate as needed to correct issues.

VERIFY QUANTITATIVELY

7. Full Rollout

You are then ready to do a full rollout. Once your feature is rolled out, it is marked “Done” and the lock of the work-in-progress limit is released. This allows you to starting working on the next feature in the backlog queue.

8. Verify Quantitatively

With the feature fully live, you should now be able to compare your conversion cohorts for the week the feature went live against the pervious week to verify the expected macro impact.

Depending on the type of feature, you might additionally need to setup a split-test. Split-testing is a matter of judgement at this stage.

The more concurrent split-tests you have going, the longer the verification time window. Long running experiments can also start interfering with other experiments and complicate your cohorts. For these reasons, it is best to use your judgement to decide when to split test versus not.

Here are some guidelines:

- I generally don’t split test brand new feature because you can compare against older cohorts that didn’t have this feature.
- I don’t split-test experiments that get very strong signals during qualitative testing.
- I do recommend split-testing experiments that got a medium to strong signal during qualitative testing and those that test improvements or alternate flows.

Have You Built Something People Want?

In this section I'll summarize the process of iterating towards Product/ Market Fit and determining when you have achieved it.

1. Review your Conversion Dashboard results weekly
Set a time every Monday to review your weekly conversion dashboard with the entire team. Identify the leakiest buckets you need to fix first.

2. Prioritize your goals and features backlog
Review your features backlog to prioritize new and existing feature improvements.

3. Formulate bold hypotheses
At this stage, avoid micro-optimization experiments. Instead come up with bold hypotheses but build the smallest thing possible to test them.

4. Add/Kill Features
Review features throughout the feature lifecycle to ensure they have a positive impact. Otherwise, rework or kill them.

5. Monitor Retention
Review your retention cohorts. Your goal is to see steady upward movement in these numbers. Otherwise, you're simply spinning your wheels.

6. Run the Sean Ellis Test
Once your retention numbers approach 40%, consider running the Sean Ellis Test.

What is the Product/Market Fit exit criteria?

You are done when you can

- retain 40% of your users,
- pass the Sean Ellis test, and
- get paid (added bonus: reach break-even)

CHAPTER 14

CONCLUSION

Congratulations! We're done.

What's Next?

Resources

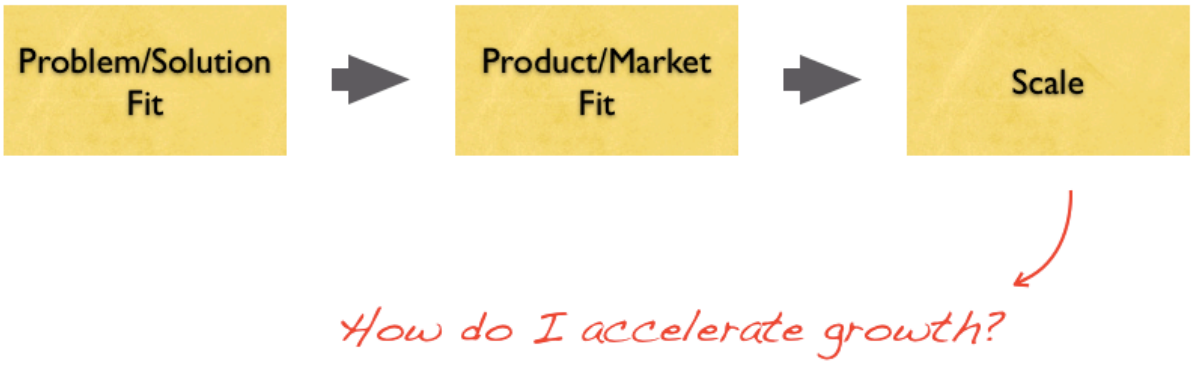
What's Next?

I believe that the life of any startup can be divided into two parts: before product/market fit (call this "BPMF") and after product/market fit("APMF").

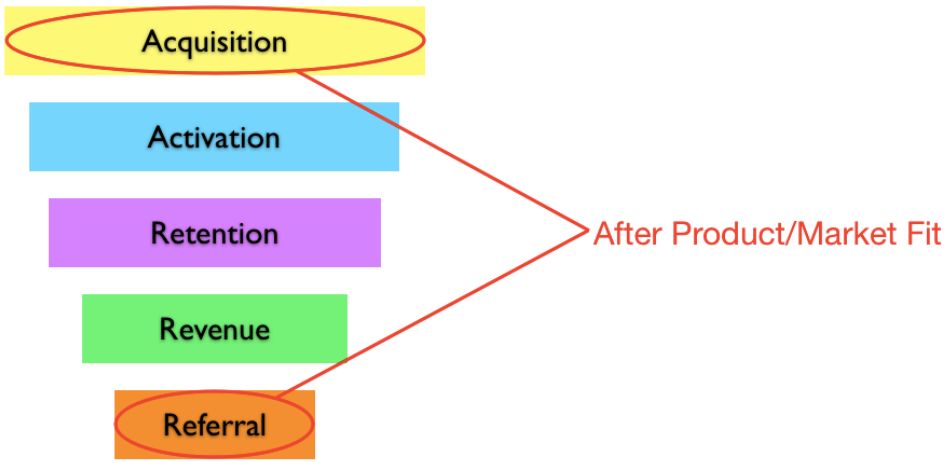
- Marc Andreessen
The Pmarca Guide to Startups

Life After Product/Market Fit

Getting to Product/Market Fit is first significant milestone of a startup. At this stage, some level of success is almost guaranteed and your focus can then shift from learning to scaling.



From a conversion dashboard perspective, after Product/Market Fit, your priority should shift from the core product features towards Acquisition and Referral (which is another form of customer acquisition).



While the meta principles presented in this book still apply, the specific tactics change. For one, quantitative metrics and split testing play a larger role in the validation process.

Did I Keep My Promise?

I started this book stating that no methodology can guarantee success but I promised a repeatable, actionable process for building products - **one that raises your odds for success while minimizing on waste**.

I hope I delivered on that promise.

There is no better time than the present to start up and the ideas in this book help you do just that. In fact you'll find that once you internalize the core principles presented here, you'll see applications for them all over the place - from writing a book, starting a blog, to even finding a co-founder.

Keep In Touch

A book, like large software, is never finished - only released.

This book is only the beginning. I will continue to share my learning on my blog: http://ashmaurya.com. And who knows, there might be more books "pulled" out of me.

I will add that you don't get a **gold star for following a process**, but **achieving results**. While I believe the meta-principles outlined in this book apply across a range of products and business models, the specific tactics most certainly can and will vary.

I have setup http://www.RunningLeanHQ.com specifically for this purpose. While this site will provide additional resources to supplement the book, my main goal is to encourage you to riff and extend on the ideas in this book as you take them to practice. The book was mainly written as "Running Lean for (mostly SaaS) Web Startups". What about "Running Lean for Marketplaces", "Running Lean for Publishing", "Running Lean for X"?

Please stop by, share your story, and join in on the conversation.

Drop me a line anytime at ash@usercycle.com or 512-524-7539.
twitter: @ashmaurya skype: ashmaurya

Thanks for reading and here's to your success!

Resources

Books

The Lean Startup - *Eric Ries*

The Four Steps to the Epiphany - *Steve Blank*

Business Model Generation - *Alex Osterwalde*r

The Entrepreneur's Guide to Customer Development - *Brant Cooper and Patrick Vlaskovits*

Positioning - The Battle For Your Mind - *Jack Trout and Al Ries*

Don't Roll the Dice - *Neil Davidson*

Rocket Surgery Made Easy - *Steve Krug*

Inbound Marketing - *Dharmesh Shah and Brian Halligan*

The Principles of Product Development Flow - *Donald Reinersten*

Lean Software Development - *Mary and Tom Poppendieck*

Toyota Production System - *Taiichi Ohno*

Blogs

Eric Ries - Startup Lessons Learned (http://startuplessonslearned.com)

Steve Blank (http://steveblank.com)

Jason Cohen - A Smart Bear (http://blog.asmartbear.com)

VentureHacks (http://venturehacks.com)

Sean Ellis - Startup Marketing (http://startup-marketing.com)

Dharmesh Shah - OnStartups (http://onstartups.com)

Tools

Lean Canvas - Business Model Validation Software (http://leancanvas.com)

USERcycle - Lifecycle Marketing Software (http://usercycle.com)

User Testing - Online usability testing (http://usertesting.com)

KISSmetrics - Actionable Web Analytics Software (http://kissmetrics.com)

mixpanel - Real-time event tracking (http://mixpanel.com)

SnapEngage - Online Customer Feedback Tool (http://snapengage.com)

heroku - Ruby platform-as-a-service infrastructure (http://heroku.com)

APPENDIX

Bonus Material

How to Build a Low-burn Startup

How to Achieve Flow in a Lean Startup

How to Build a Low-burn Startup

I've bootstrapped my company for the last seven years and learnt a lot about bootstrapping from Bijoy Goswami, founder of Bootstrap Austin. Bijoy doesn't limit the definition of bootstrapping to the more commonly held one about *building a company without external funding* but rather views bootstrapping as a philosophy summarized as **"Right Action, Right Time"**.

This mantra applies just as well to Lean Startups as it does to Bootstrapped startups:

At every stage of the startup, there are a set of actions that are "right" for the startup, in that they maximize return on time, money, and effort. A lean/bootstrapped entrepreneur ignores all else.

While Bootstrapping and Lean Startup techniques are not just limited to funding, funding is one of the first problems entrepreneurs tackle which can lead to waste.

Why Premature Fundraising is Waste

Getting funded is not validation

Seed stage investors are just as bad at guessing what products will succeed as you are. Without any product validation to rely on, they hedge their bets against your team's past track record and storytelling ability. So while getting funded at this stage is a testament to your team building and pitching skills, it isn't product validation.

Without validation you have no leverage

More importantly, without validation you don't have product/market credibility which typically comes at a price – reflected in lower valuations and investor-favored term sheets.

Investors measure progress differently

While validated learning is the measure of progress in a Lean Startup, most investors measure progress through growth. Reconciling the two during the early stages of a startup (when the hockey stick is largely flat) can be both challenging and distracting.

Getting funded always takes longer than you think

Time is more valuable than money. Would you rather spend months pitching investors so you can refine a story based on an untested product, or spend time pitching customers so you can tell a credible story based on a tested product?

Too much money can actually hurt you

Money is an accelerant, not a silver bullet. It lets you do more of what you're currently doing but not necessarily better. For instance, more money might tempt you to hire more people and build more features - both of which may lead you off course and slow you down.

Constraints drive innovation but more importantly force action.

With less money, you are forced to build less, get it out faster, and learn faster.

What about all the advice and connections?
Raising funding is not the only way to get good advice. You can and should start building a diverse board of advisors early. Many are happy just to be asked, others might require a little equity to formalize a relationship.

How do I Survive Until Product/Market Fit?

While the ideal time to raise external funding is after Product/Market Fit, you might need to raise a smaller round before then or self-fund. The goal is to get as close to Product/Market Fit as possible.

The biggest reason for bootstrapping first is that it is easier than ever to start a company or more accurately test to see if you even have a company. You don't need much to start defining, building, and testing a minimum viable product (MVP) towards product/market fit. With the right team (and skill-set) in place, you can validate Problem/Solution fit while keeping your day job and put a MVP in front of customers soon after that.

Here are some other techniques to help you along the way:

Keep your day job

The first stage, finding Problem/Solution fit, can really be done part-time with very little burn. It typically has a lot of waiting time built-in e.g. contacting customers, scheduling interviews, collecting results, etc. Until you find a problem worth solving, it really doesn't make sense to quit your day job. The outcome of this stage should be a handful of features.

Build just those features, and nothing else.

While all this can usually be done in your spare time I highly recommend reviewing your company's moonlighting clause.

Disclaimer: I am not a lawyer. You should consult a lawyer before applying this to your particular situation.

Conserve burn rate

The biggest burn in a software business is people. Hardware is cheap. Rent don't buy. Don't scale till you have a scaling problem. Don't hire till it hurts.

Charge from day one

Make a goal of first covering your hardware/hosting costs, then your people costs.

Sell other related stuff along the way

It is very tempting to take on unrelated consulting to survive but it becomes very hard (if not outright impossible) to build a great product in parallel. Instead look for other related stuff you can sell along the way. License out a piece of your technology, write a book (like this one), teach workshops, get paid to speak, etc. Not only are these things related to your core business, but many of them also help you build up your online reputation and brand which pay off over time and could even lead to an unfair advantage.

Bootstrapping + Lean Startup = Low-burn Startup

While not the same thing, bootstrapping and Lean Startups are quite complementary:

Bootstrapping provides a strategic roadmap for achieving sustainability through customer funding (i.e. charging customers) and Lean Startup techniques provide a more tactical approach to achieving those goals through validated learning.

How to Achieve Flow in a Lean Startup

Eliminating waste is a fundamental principle from “Lean”.

Waste is any human activity which absorbs resources but creates no value.

- Womak/Jones
Lean Thinking

Of all resources, there is no resource more valuable than time. Time is more valuable than money. While money can fluctuate up or down, time only moves in one direction.

The Conflicting Pull for Time

Time, like any resource, has multiple pulls. In following customer development there is a basic pull for activities outside the building versus inside the building. Steve Blank asserts that all the answers lie outside the building and advocates the creation of a cross-functional customer development team **which must include the founders**. What about work that needs to get done inside the building? Who is going to implement the solutions to problems uncovered outside the building?

Eric Ries’ answer is to create 2 teams that feed each into other: a problem team and a solution team. The first team focusses on customer development, while the second team focusses on product development.

However, if you are a founder, you need to be on both teams wherein lies the fundamental **scheduling tug-of-war**.

The problem is further exacerbated if you are a technical founder (like me) because time is utilized very differently when switching from product development to customer development. Paul Graham wrote an excellent essay on the two types of schedules: manager's schedule and maker's schedule.

Managers typically organize their day into 1 hour blocks, and spend each hour dealing with a different task. Makers, like programmers and writers, need to organize their day into longer blocks of uninterrupted time. The cost of context switching is low (and expected) in a Manager's schedule. It is high (and a productivity killer) in a Maker's schedule.

Activities outside the building (customer interviews, usability testing, customer support) tend to be on a Manager's schedule while activities inside the building (design, coding) are usually on a maker's schedule.

Trying to find an equilibrium point between these two pulls is more art than science but there is fundamental concept that must be present to maximize productivity – **flow**.

There are 2 different definitions to what I mean by flow and both apply here.

The first comes from psychologist Mihály Csíkszentmihályi who defines flow as a mental state of operation when you are at your best. When you are in flow, you are so totally immersed in an activity that nothing else matters. You loose your self-consciousness and sense of time.

Activities that flow typically have the following attributes:

- Have a clear objective
- Need your full concentration
- Lack of interruptions and distractions
- Clear and immediate feedback on progress towards objective
- A sense of challenge

While flow can't be triggered at will, you can arrange activities so they allow for flow which coincidently is also the second definition of flow.

When we start thinking about the ways to line up essential steps to get a job done to achieve a steady continuous flow with no wasted energy, batches, or queues, it changes everything including how we collaborate and the tools we devise to get the job done.

- Womak/Jones
Lean Thinking

What follows are specific work hacks I use to allow for flow:

Creating Daily Flow

I generally group my daily activities into 3 categories: Planned maker activities, planned manager activities, and unplanned maker/manager activities.

Work Hack 1: Establish uninterruptible time blocks for maker work.

My planned maker activities are typically coding and writing tasks I've previously identified. Because these activities need an uninterruptible block of time, I schedule these very early in the morning (6am-8am). I usually schedule this task the night before and it is the first and only thing I do. I

don't check email/twitter or look at anything else. No one is calling at that hour so distractions are at a minimum. I find two hour blocks work best for me.

Work Hack 2: Achieve maker goals as early in the day as possible.

I've tried both staying up late and waking up early, and prefer the latter as it isn't interrupted by sleep which allows the day's activities to flow better. I also personally find that accomplishing something tangible that early in the day sets the tone for the rest of the day.

Depending on the day of the week, I might allocate more 2-hour blocks later in the morning or afternoon but they aren't as intense as the first one and can be interrupted by something more urgent.

Work Hack 3: Schedule manager activities as late in the day as possible.

Planned manager activities, like customer meetings, are easier to schedule because they are clearly time-boxed and calendar driven. Unless there is an unworkable schedule conflict, I prefer to schedule these for the afternoon so as not to interrupt my morning flow.

Work Hack 4: Always be ready for unplanned activities like customer support.

Unexpected interruptions can surface from anywhere throughout the day – server issues, customer support calls, etc. You have to be prepared for interruption, especially from customers. Both server alerts and customer calls (800 number) are routed directly to my mobile phone. This is a also

good place to apply a Five Whys process to ensure unexpected incidents don't become recurring.

Creating Weekly Flow

Aside from organizing the day for flow, I also group certain activities of tasks by day of the week.

Work Hack 5: Identify the best days for planned customer development.

For instance, Mondays and Fridays are usually bad days for initiating new customer contact as people are generally either recovering from the weekend or getting ready for it. I plan these types of customer development activities between Tuesday-Thursday.

Work Hack 6: Take advantage of customer downtime.

Since Mondays and Fridays are usually slower from a customer perspective, I use them for larger maker tasks like writing blog posts. My blog posts are usually identified on Friday, outlined roughly over the weekend, written/proofed on Monday, and published on Tuesday.

Work Hack 7: Balance face time with customers.

Not all customer development activities require face time. Beyond the initial customer discovery stage, there is a strong tendency to rely more heavily on asynchronous communication using tools like email, forums, and online usability testing. While all these tools are great for lowering real-time distractions and achieving scale, I find it important to still create opportunities for face time with existing and new customers.

Unscripted conversations are the best way for learning about unscripted problems.

I put our 800 number on all pages and encourage customers to pick up the phone versus email whenever possible.

Eliminating software waste

Building software to specifications is hard enough that when faced with a startup environment where both problems and solutions are largely unknown, it is optimal to iterate around less code and more learning.

Work Hack 8: Avoid overproduction by making customers pull for features.

Customer pull is another concept from "Lean" and it requires that no product or service be produced until a customer asks for it.

80% of your effort should be spent towards optimizing existing features versus building new ones.

The whole point of customer development is identifying an MVP that resonates with customers and the whole point of customer validation is testing if that resonance will scale. If it doesn't, the answer is not adding features, but possibly pivoting and going back to step 1 – customer discovery.

Work Hack 9: Iterate around only 3-5 actionable metrics.

Having a few actionable metrics is all you need to identify and prioritize the most critical issues to tackle.

Work Hack 10: Build software to flow.

You might have noticed I don't have days or tasks identified for building, testing, or releasing software. That is because I follow a continuous deployment process (also popularized by Eric Ries) where software is built, tested, and packaged automatically at the end of every maker task with no effort on my part other than checking in code. One-click and the code is released to customers.

Manufacturing processes have traditionally arranged around machine time breaking tasks into batches and queues. "Lean" challenges this approach and calls for arranging around human time organizing tasks so they flow.

Releasing software is not unlike manufacturing. While it is somewhat easier to continuously deploy web based software, with a little discipline, desktop based software too can be built to flow.