Start Small, Stay Small: A Developer's Guide to Launching a Startup

Rob Walling

Start Small, Stay Small

A Developer's Guide to Launching a Startup

Written by Rob Walling

Edited by Mike Taber

www.startupbook.net

ISBN 978-0-615-37396-6

First Edition

For my Mom and Dad, who years ago taught

me the value of hard work.

Contents

Preface

Who is this Book For?

This book is aimed at developers who want to launch their startup with no outside funding. It's for companies started by real developers solving real pain points using desktop, web and mobile applications.

This book intentionally avoids topics restricted to venture-backed startups such as: honing your investor pitch, securing funding, and figuring out how to use the piles of cash investors keep placing in your lap.

In this book I assume:

- You don't have $6M of investor funds sitting in your bank account
- You're unable to relocate to the handful of startup hubs in the world
- You're unwilling to work 70 hour weeks for low pay with the hope of someday making millions from stock options

There's nothing wrong with pursuing venture funding and attempting to grow fast like eBay, Google, Twitter, and Facebook. It just so happened that most people are not in a position to pursue this option.

What Does This Book Cover?

The focus of this book is building and launching a successful software, web or mobile startup with no external funding.

This process includes:

- Developing the proper mindset for a self-funded startup
- Understanding the Market-First Approach
- Finding and testing a niche market
- Choosing the optimal platform, price and revenue model
- Building a killer sales website
- Understanding the primary purpose of your sales website
- Building the right kind of interest, and thus driving the right kind of traffic, to your website
- Learning how to outsource
- Working with virtual assistants
- Determining what to do after launch: do you grow the business or start over?

If you're reading this book I assume you are an experienced software developer, so we won't be covering the development process. If you need assistance building software there are books written with that purpose in mind.

As I began the writing process, I received input from developers who told me they were tired of "filler" material – weak case studies, superficial interviews, chapters describing 10 options for how to accomplish a task but no guidance on which path to take or how to decide between them.

So I've focused on providing a practical, step-by-step approach to getting your startup off the ground and focused on making every word count. No filler allowed.

If you aren't frantically underlining, highlighting or taking notes as you read each chapter, I have not achieved my goal for this book.

To ask a question, make a comment or purchase the print, ebook or audio version of this book, visit www.StartupBook.net

About the Author

One question that should be on your mind as you start reading any book is: why should I listen to this author? Here is my track record so far.

My day job is managing 10 profitable software products and websites that I've built or acquired over the past ten years. I blog about startups, online marketing, and software and web entrepreneurship at www.SoftwareByRob.com.

And with my colleague Mike Taber, I run an online startup community with several hundred members called the Micropreneur Academy.

About the Micropreneur Academy

The Micropreneur Academy is a paid online learning environment and community website for startup founders.

Although this book contains a small amount of material from the Micropreneur Academy, it only covers topics that are conducive to the printed page.

The purpose of the Micropreneur Academy is to present topics that require interactive elements (screencasts, audio, and worksheets), topics that change frequently, cutting-edge approaches and our complete Rolodex of vendors and contractors. In addition it provides a community of like-minded startup founders accessible via private forums.

If you are interested in launching or growing your startup faster, as an owner of this book you are entitled to the first month of the Academy at no charge. To receive your free month visit www.Micropreneur.com/book/

And with that…let's get started.

Chapter 1

The Chasm Between Developer and Entrepreneur

What is an Entrepreneur?

An entrepreneur is a visionary.

He is the person who sees the potential in an idea and creates a viable business from nothing.

She is the person who invests hundreds of hours into building, launching and marketing a product, fighting through every roadblock along the way.

In this book we'll cover the approaches of two types of entrepreneurs:

- **Micropreneurs** – Entrepreneurs who want to remain solo. This means no employees under any circumstance. Micropreneurs might own a single product, or may own many products that collectively contribute to their bottom line. A specific lifestyle is the goal of a Micropreneur.

- **Bootstrappers** – A bootstrapper has their vision set on something larger than a single person venture. Perhaps 5 employees, perhaps 10…a bootstrapper has an idea and knows she must execute and grow her company to survive.

It so happens that 90% of the knowledge needed to succeed at Micropreneurship and bootstrapping is identical. Finding a niche, finding a product, building, launching, marketing…it's the same process.

In this book I will point out a handful of places where the two paths diverge and call out the recommended approach based on the path you're following.

Why the Anti-Venture Capital Stance?

I wanted to address this early so you're clear on the focus of this book.

I am not anti-venture capital. I am anti-everyone-thinking-venture-capital-is-the-only-way-to-start-a-tech-company.

Seeking funding of any kind creates two problems.

First, it involves a massive investment of time and focus, which distracts you from the important things, like making money and staying in business.

Second, it makes modest success nearly impossible due to the limits it places on the potential markets you can pursue.

If you're self-funded with one or two founders, you can support your entire business from a tiny niche that provides $10k/month in revenue.

But with venture capital (or even a few hundred thousand in angel funding) you are forced to go after much larger markets. And a market that's 10x the size is 100x harder to get right. There's more competition, more complexity, higher advertising

rates, more SEO competition, and a more fragmented audience.

Targeting a large, non-niche market is expensive in terms of marketing and support. It will eat you alive if you tackle it from the start.

But if you start small and make a product so good that your niche is falling all over itself to sign up, word will spread and you will soon find yourself with a product that extends beyond your tiny niche.

However, this takes time to grow organically; an approach that outside funding does not allow.

A Look at the Self-Funded Startup Entrepreneur

There are many definitions of entrepreneur, but since we'll be discussing self-funded tech startups we are going to focus our definition on a few key points:

Point 1: An entrepreneur is a technical visionary who creates software for a niche market.

Niche markets are critical. If you want to self-fund a startup you have to choose a niche.

Building an online invoicing software as a service (SaaS) application? Good luck.

Building an online invoicing application targeted at landscape architects? Now you're talking.

The genius of niches is they are too small for large competitors, allowing a nimble entrepreneur the breathing room to focus on an underserved audience. Once you've succeeded in that niche, you can leverage your success to establish credibility for your business to move into larger markets.

Point 2: An entrepreneur merges existing technical knowledge with online marketing knowledge.

The key factor in an entrepreneur's success is their ability to market their product. I can almost hear you groaning…but keep an open mind. Millions of people in this world can build software. A fractional subset of those can build software *and* convince people to buy it.

A developer who knows how to market a product is a rare (and powerful) combination.

Point 3: An entrepreneur is a cross between a developer, a webmaster, and a marketer.

Developer
Software entrepreneurship would be nearly impossible without the technical skills we learn as developers. The ability to fix a production site that's crashing or put together a hot fix for a key customer will be critical to your success.

Webmaster
These days it's imperative that your startup knows how to sell online. This means creating a website that converts visitors into customers. We'll be exploring a number of ways to build high-converting sales websites, and doing these will require basic HTML knowledge.

Once your website is up, you will be making constant updates, adding new content to achieve better search engine rankings, tweaking conversions (newsletter sign-ups, trials, purchases, etc…) and most of these tasks will be better executed if you can make small changes yourself. Basic knowledge of HTML or a CMS is essential.

Marketer
Marketing is more important than your product.

Let me say it another way:

> *Product Last. Marketing First.*

Your product has to be good. If it's not, you'll be out of business.

But before you build a good product you have to find your market. With an enormous amount of anecdotes to back me up I strongly believe that building something no one wants is the most common source of failure for entrepreneurs.

In chapter 2 we'll look at how to make sure people want what you're building *before* you build it.

Self-funded vs. Venture-funded

Self-funding your startup is worlds apart from what most people talk about in the tech startup world.

Venture-backed startups typically raise capital from outside investors with the goal of massive, accelerated growth. They attack large, growing markets with the hopes of 100x growth in 5 years or less.

Examples abound, but think along the lines of Netscape, Google, Travelocity, Yelp, Twitter and Facebook. Big dollars. Fast growth. High risk.

If you're a venture-backed startup founder you're looking at many years of long hours with a small potential for a huge payoff.

A bootstrapper still seeks growth, but at a slower pace since they do it organically through the re-investment of profit. They

focus on smaller markets or niches within larger markets and grow both profit and headcount organically.

Examples include Joel Spolsky's Fog Creek Software and Eric Sink's SourceGear. Keywords: slower growth, less risk.

If you're a self-funded startup founder, you're looking at a decent potential for a decent payoff.

A Micropreneur has goals that are not limited to financial gain and more often involve lifestyle goals. Being able to live where you want and work when you want is an appealing option for many founders.

Two Real-life Examples

A Micropreneur: Ruben Gamez
Ruben Gamez is a Micropreneur. He has a full-time job as a development manager, but launched his SaaS application Bidsketch[1] in 2009. Today, he makes a tidy monthly income from this application. Ruben has no plans to leave his job, but enjoys the extra income, the control he can exert over his own project, and the experience of learning how to build, launch and market his own application.

The lessons Ruben has learned over the past year of launching and growing Bidsketch will apply to his future efforts, and he plans to launch other products during the coming year. His long-term goal is serial Micropreneurship and eventually, self-employment.

A Bootstrapper: Harry Hollander
Harry started Moraware Software[2] with a business partner. Their goal was to provide scheduling software for countertop installers, and within six months they knew it was a viable

[1] www.bidsketch.com

[2] www.moraware.com

business. Eight years later Moraware employs four people in addition to the two founders, all of whom work virtually from their respective homes.

Moraware focuses on the construction vertical niche and more specifically, countertop contractors. Limiting themselves to this group of people has a lot to do with their success. They've been able to focus on filling the exact needs of their customers and as a result they own a large portion of the countertop scheduling software market.

Starting for the Right Reasons

Most developers want to build software products for the wrong reasons.

Reason #1: Having a Product Idea

If you have an idea for a product, odds are high that you have *project/product confusion*.

A *project* is a software application that you build as a fun side project. The code is fun to write because you're not concerned about quality and performance, and the end result is a neat little application that likely isn't of use to many people.

A *product* is a project that people will pay money for. In other words, it's a project that has a *market* (a group of people who want to buy it). Without a market, a software application is just a project.

Most developers who come up with an idea know exactly how they will build it, but no idea how they will reach potential customers. They think a link from TechCrunch will drive hundreds, if not thousands, of sales.

On rare occasions the product-first approach works, but for the most part it's a recipe for failure.

Reason #2: To Get Rich

Getting rich shouldn't be your goal when launching a product.

Thousands of people with significantly more coding and marketing skills have built and launched products, yet still work for a living. If you are doing it for the money you will not stick around during those long months of hard work when no money is coming in.

If you want to make a million dollars, buy a lottery ticket or start a venture-backed startup; the odds of succeeding at either are very much not in your favor, but the potential payout is big.

As I said earlier, I'm pretty sure you're here to do something with less risk and less reward, but a much higher chance for success. A million dollar payday is most likely not in your future, but owning a successful startup can be.

Reason #3: Because It Sounds Like Fun

We've all read the stories of a successful startup and dreamed that we would one day be that person in the *Fast Company* article. Whether it's someone who hits it big with a fluke Facebook application or works for years to build a successful software company that you've followed through their blog, the story is always romanticized. In other words, the few glimpses you have into the life of a startup are not a true indication of what goes on behind the scenes to make it work.

If you want to become a Micropreneur because it *sounds like fun,* you're going to have a rude awakening on the 10th day when the initial excitement has worn off and you're slogging through exception handling code at one in the morning.

To throw more fuel on the fire, what you know about software development is a small piece of the puzzle. Writing code, where most of us are well-versed, is only about 30% of the work needed to launch a successful product.

The other 70% is debugging, optimizing, creating an installer, writing documentation, building a sales website, opening a merchant account, advertising, promoting, processing sales,

providing support, and a hundred other things we'll dive into in later modules. Some of it is great fun…other parts, not so much.

Suffice to say that being an entrepreneur can be fun, but the fun parts come only with hundreds of hours of hard work.

What are the Right Reasons?

The "right" reason to start a startup depends on your goals. As I mentioned before, Micropreneurs lean towards lifestyle choices (freedom, income independence, location independence), while bootstrappers might embrace the challenge and excitement of owning their own business, to build equity in something they own, and to have control over the projects they work on.

Paying the Price of Success

Software entrepreneurship is a fantastic experience. The first time someone pays you for software you wrote, your head will nearly spin off its axis.

And ultimately, if you're able to harness the power of leveraging software instead of time, you can achieve more freedom than you're ready for. The first time I took a one month vacation I had the nagging feeling that I needed to do 160 hours of work when I got home…then I realized that hours and dollars no longer correlated.

It's hard to re-train your mind out of the dollars-for-hours mentality. For me it took well over a year.

The price to achieve this kind of payoff involves a huge up-front investment of time.

Or it may involve a substantial financial investment acquiring products.

Or maybe you'll decide to step away from the code, give up that control we all feel we need, and hire out some development so you focus on other areas of the business.

The price you pay is negotiable and is truly guided by your personal goals. But the bottom line is: you will have to pay a price.

It's a long road to becoming a successful entrepreneur. There will be many long nights, especially in the beginning. It's critical to know what you want out of entrepreneurship so you can make the right decisions along the way, and to give you something to hold onto when you're burning the midnight oil for the fifteenth night in a row.

What Are Your Goals?
Knowing your goals will allow you to make the right decisions as you start (or continue on) your path through entrepreneurship.

People start startups for a plethora of reasons, both personal and professional. Hard decisions lie ahead and the answers depend on what you want to get out of your startup experience.

As an example, which of the following sounds most appealing?

1. Keeping your day job and earning extra money on the side.
2. Building a portfolio of products and quitting your day job while still writing code.
3. Quitting your day job with the intent of outsourcing the code and focusing on the entrepreneurial side.
4. Quitting your day job and running a small company where you work from home and may have a few employees.

There are other options, of course, but your answer to the above depends on your personal preferences, desires and time constraints. Luckily you don't have to decide everything now.

But it's good to begin thinking about your motivation for starting a startup because it dictates the type of product you should launch.

The Power of Goals

To nail down what you want out of entrepreneurship, you need to decide on your goals. This is a process most people skip due to skepticism about the benefits of the process.

A study at Dominican University[3] revealed that the following 3 factors substantially increased someone's chance of following through on their goals:

1. **Written Goals** – "Those who wrote their goals accomplished significantly more than those who did not write their goals."
2. **Public Commitment** - "…those who sent their commitments to a friend accomplished significantly more than those who wrote action commitments or did not write their goals."
3. **Accountability** – "…those who sent weekly progress reports to their friend accomplished significantly more than those who had unwritten goals…"

It may feel like you're an exception; that you don't need goals or accountability…but trust the science and give it a shot. Spend 20 minutes making a list of the things you are hoping to accomplish by starting up. If you believe what was said above, it will make a big difference. Worst case, you waste 20 minutes of your time.

Remember that there is no single *best* path to success as a startup founder. Since you are deciding on a specific lifestyle and are making sacrifices to get there, it can look like almost anything. Just be sure it's what you want.

---

[3] Summary of Recent Goals Research, by Gail Matthews, Ph.D., Dominican University

The second two items, public commitment and accountability, can be achieved by interacting with a community of like-minded startup founders. You can find this through local Meetup groups[4], or through an online community like the Micropreneur Academy[5].

In the Academy, we have an accountability room where founders post weekly updates on their progress and discuss roadblocks that are impacting their progress because we know that it does help people to achieve their goals.

One Short-Term Goal I Propose

Here's one question you should think about right now: what is a good short-term goal for your startup?

I have a suggestion to help get you started:

> *Strive to build a startup that generates $500 per month in profit.*

This may sound like an easy goal, but will require more work than you can fathom at this point.

Once you've done that you will have so much experience under your belt you won't believe how much you know (and how little you knew when you started).

The Dip (a.k.a. The Software Product Myth)

The reason you need goals and accountability is to stay motivated during the hard times. Without goals you are much more likely to throw in the towel when things get difficult. This might be when you launch and no one buys, or you are

[4] www.meetup.com

[5] www.micropreneur.com

bombarded with so many support requests you don't have enough time to build new features.

Most developers start as salaried employees, slogging through code and loving it because they never imagined a job could be challenging, educational, and downright fun. Where else can you learn new things every day, play around with computers, and get paid for it?

A certain percentage of developers become unhappy with salaried development over time (typically it's shortly after they're asked to manage people, or maintain legacy code), and they dream of breaking out of the cube walls and running their own show. Some choose consulting, but many inevitably decide to build a software product.

"After all," they think "you code it once and sell it a thousand times – it's like printing your own money! I build apps all the time, how hard could it be to launch a product?"

Against All Odds

Most often the developer who chooses to become a consultant (whether as a freelancer or working for a company), does okay. She doesn't have a ton of risk and gets paid for the hours she works.

But developers who make the leap to their own startup are another story. Building a product involves a large up-front time investment, and as a result is far riskier than becoming a consultant because you have to wait months to find out if your effort will generate revenue. In addition, growing a product to the point of providing substantial income is a long, arduous road.

But let's say, for the sake of argument, that you spend 6 months of your spare time and you now own a web-based car key locator that sells 100 copies per month at $25 a pop. At long last, after months of working nights and weekends, spending every waking moment poring over your code,

marketing, selling, and burning the midnight oil, you're living the dream of owning your own startup.

Except for one thing.

Support is Brutal

In our scenario you're now making $2500/month from your product, but since you make $60k as a salaried developer, you're not going to move back in with your parents so you can quit your day job.

So you work 8-10 hours during the day writing code for someone else, and come home each night to a slow but steady stream of support emails. The worst part is that if you've built your software right the majority of the issues will not be problems with your product, but degraded OS installations, crazy configurations, a customer who doesn't know how to double-click, etc…

The next step is to figure out, between the 5-10 hours per week you're spending on support, and the 40-50 hours per week you spend at work, how you're going to find time to add new features. The kicker is the support burden actually worsens with time because your customer base grows. After 1 month you have 100 customers with potential problems, after a year, 1,200.

And yes, the person you decided to sell to even though they complained about the high price ($25) is still hanging around, emailing you weekly wondering when the next release is coming.

But you persevere, and manage to slog your way through the incoming support requests and get started on new features.

What you find is that ongoing development, as with any legacy system, is much slower than green field development. You're now tied to legacy code and design decisions. You soon realize this isn't what you signed up for when you had that brilliant

flash of insight that people need web-based software that helps them when locating their keys.

It's about this time that support emails start going unanswered, releases stop, and the product withers on the vine. It may wind up for sale on Flippa (www.flippa.com), or it may be relegated to the bone yard of failed software products.

The Upside
The flip side is what you've already heard on the blogs of successful product developers.

Once a product hits critical mass, you've conquered the hardest part of the equation. After that, the exponential leverage of software products kicks in and you can live it up on your empire of web-based unlocking-device locator applications. It's a recurring revenue stream that can grow far beyond what you would make as a consultant, all the while creating balance sheet value (meaning one day, you can sell it for stacks of proverbial cash and retire).

This is unlike your consultant buddy, whose consulting firm is worth 44 cents once she decides to retire because she had an unused stamp on her desk.

But there is a dip before you get to this place of exponential leverage and proverbial cash. A big dip. And if you can get through it once, it's more likely that you'll be able to get through it with your next product. And the one after that.

The key factor in getting you through the dip is your goals. These are the goals you wrote down six months prior due to the advice given to you in a random book about startups.

Once you make it to the other side, you've learned what it takes to launch and maintain a product. The next time you launch a product, you will have a monumentally better chance of success because you are now a more savvy software entrepreneur.

Why People Make the Switch from Developer to Entrepreneur

As a developer, your income, your ability to control what projects you work on, and your ability to control what you learn, end at a certain point.

For the first several years you're constantly learning, working on (seemingly) fun projects and your income grows quickly if you apply yourself.

But the experiences of many veteran developers show that after a certain level of experience you can't push past that financial barrier. And keeping up with the latest and greatest technology – something that used to excite you – will begin to wear you down.

It's hard to break out of that position and most often it requires a big risk. In my experience this risk involves joining an existing startup or starting one of your own.

I've tried both. I did it more for the excitement and freedom than for the income. In fact, I took a pay cut when I moved from consulting to owning my own products. But my passion for building something I own and the opportunity to experience true time and location independence far outweighed the drop in income.

Lack of Learning

Part of your "topping out" will likely be a lack of learning. While it's true there are always new technologies to learn, as you mature in your technology career it's likely you will begin to feel like a hamster on a wheel as you learn one more way to pull data out of a database. The idea of spending the time to learn how to do it using the latest method begins to make you tired.

It starts to feel as if you are constantly moving from one technology to the next which involves little "real" learning and

a lot of learning some new syntax or API…things that will change in six month anyway. You start to feel like learning new techniques for the rest of your life will basically rehash the same things you learned in your first two years as a programmer.

Ownership
I like to use the analogy of renting vs. buying a home. When you rent you have less commitment, you can move often, and renting is typically less expensive than buying. But you don't own anything and you don't build equity. When you move out you're not any better off than when you moved in.

Such it is with salaried employment and consulting. When the day is done you own nothing. Not only does this translate into a lack of financial gain, it's a mental challenge as well. Many find it hard to build application after application and never feel passionate about the application they're building.

Once you launch a product, you are instantly building equity. With every copy you sell and every improvement you make, you are building something that will not only generate income in the future, but actually has value on the open market. Instead of having nothing at the end of your lease, you wind up getting back all of your rent money and more in equity.

The Biggest Roadblocks to Your Success

There are many roadblocks between the day you decide on a product idea and the day you launch. Your day job, family commitments, or the allure of the TiVo have a tendency to chip away at your grand ideas and leave you feeling overwhelmed, unproductive and unmotivated.

After 6 months of building your product with little more to show than a few thousand lines of plumbing code, it's easy to lose focus and shut the whole thing down.

The biggest roadblocks I've experienced first-hand or discovered through conversations with entrepreneurs are discussed below, along with strategies for how to avoid them.

Roadblock #1: No Market
This is by far the most common mistake I've seen – building something no one wants. We'll discuss the importance of having a market in more detail in chapter 2.

It's a common belief that building a good product is enough to succeed. It's not.

If you are the typical developer, you will not have enough money, power or fame to generate demand for a product people don't need or don't know they need.

How to Avoid It
Avoid this roadblock by building a product *after* you've verified there is a market.

In chapter 2 we'll look at the right questions to ask about a potential market, how to answer to those questions using research, and how to test the validity of your market for a minimal investment before you write your first line of code.

Roadblock #2: Fear
The first time you try something it's scary.

As humans we fear the unknown. We fear failure…rejection…mistakes. These are either feelings that come naturally or have been pressed into us by society.

This fear is what makes starting a company hard. It entails a large amount of risk with so much potential for failure, rejection and mistakes.

How to Avoid It

While there's no way to avoid the fear of starting your company, the following can help put the fear in perspective:

> *The up-front fear is a big indicator that you're going to grow as a person if you proceed through it. And, frankly, the terror wears off pretty quickly.*

It's true. Surprisingly, anything is much easier the second time. And the third. And by the fourth time you can't even feel the hair on the back of your neck, or the sweat in your palms because it's no longer there. The terror goes away surprisingly quickly.

The interesting thing is that the more you taste this growth, the more you want it. It's a rush, and it's addictive.

It's a huge confidence boost to look back at the things that scared you last month, last year, or even five years ago. And realize you conquered them.

This kind of success leads you to trust your instincts. It builds confidence. It eliminates pointless analysis that used to keep you spinning on decisions for hours, days or weeks.

Overcoming the terror of firsts is hard, but it's what makes the goal beyond it worth achieving. The terror that stands between you and the goal is something 99.9% of people will never overcome.

Roadblock #3: Lack of Goals

Having no clear, written goals for your startup means you won't know whether to pursue the white label deal someone offers you two weeks after launch, or to start selling in overseas markets because someone asks you to.

Without goals for both yourself and your startup you are flying blind without guidance in situations where there is no right or

wrong answer. Answers need to stem from your long-term desires for your startup and yourself.

- Want to grow as large as possible? Make that your goal and take advantage of every opportunity that comes your way. Realize this will mean hiring employees, and working longer hours.
- Want to spend more time with your family and quit your 9 to 5? Make that your goal and realize you will have to turn down many opportunities that come your way.

Your goals must serve as your roadmap that takes you to your definition of a successful startup.

How to Avoid It
You've already heard this a few times, but define your goals and *write them down.*

Roadblock #4: Inconsistency
The main problem with inconsistency is that it makes you lose momentum and momentum is critical to staying productive.

Forget the TV and video games. How many times have you found yourself thinking you were being productive only to look back and realize you spent 3 hours searching for and evaluating something you may not need until 6 months down the road?

Spending an evening finding 50 blogs to market to is a great way to feel productive, but do you really need 50, or could you get by with 10? Shouldn't you have outsourced this task for a pittance to any respectable virtual assistant (VA)? In reality, the 4 hours you spent researching should have been 10 minutes spent writing up this task.

The hard part is that it sure feels productive to spend 4 hours doing research and it's fun, to boot. Unfortunately, it doesn't get you closer to launching.

Another common distraction masquerading as productivity is reading business books. It sure seems like *Why We Buy*[6], *Made to Stick*[7] and *Outliers*[8] are going to help you launch a successful product. But reading books gets you no closer to launching than watching *Lost*.

If reading business books is a hobby, fantastic. But it won't get you one hour closer to launch.

How to Avoid It

You've likely heard of the concept of an information diet.

The idea is that most of the information we consume is a waste of time. Newspapers, magazines, blogs, podcasts, the news…are all enjoyable to consume, but they have a tendency to offer a constant distraction from real productivity.

You can't consume and produce at the same time – when you're in high-producing mode you have to temporarily step away from your magazines, blogs, and other forms of distraction for a while. Being in the pattern of checking your RSS reader every time you sit down at your computer kills hours of productivity each week. Those are hours that could be spent building your product.

It's not easy, but scaling back your information consumption will have a huge impact on your productivity.

Start by checking your RSS reader once a day and limit yourself to 30 minutes.

Limit your news reading and TV watching to X minutes (whatever you're comfortable with).

---

[6] http://tinyurl.com/23bxxtb

[7] http://tinyurl.com/2b8v2x2

[8] http://tinyurl.com/23w2oz7

Anytime you're on your computer ask yourself "Is this activity getting me closer to my launch date?"

Roadblock #5: Believing You Have to Do Everything Yourself

I'm in the process of buying a house. The house we made an offer on last week has 45 images I would like to share with family and friends, but they are hidden behind a questionable JavaScript interface. Using my advanced knowledge of web hackery (i.e. View Source), I grabbed the list of each image URL and put them in a text file. The following ten seconds made a huge difference in how I spent the next 20 minutes of my day.

I copied the first URL into my clipboard and began to paste it into my address bar when I (for the hundredth time) realized that this is exactly the kind of task that appears to produce something, but is completely rote and repetitive. It would be simple (and fun) to write a Perl script to do the fetching, but that would take around the same amount of time.

So I sent the task to my virtual assistant (VA). It took me exactly 90 seconds to get the request to him and within 24 hours, I had a zip file of the images. It took 20 billable minutes and at $6/hour the 24-hour wait was well worth it.

This morning I realized I needed an image for one of my websites and a change to the CSS. I'm not a great designer but I could have designed something in about two hours. I could have also made the CSS change and tested it in a few browsers in about an hour.

Instead, I opted to send these simple tasks to someone else at the cost of $15/hour. I wrote up an email and the task will be done in the next day or two.

- Time spent: 10 minutes
- Time saved: 2 hours, 50 minutes

How Much Can You Really Gain?

These are trivial examples of what I call *drip outsourcing*; outsourcing small tasks as I perform my daily work. Drip outsourcing has become invaluable to my productivity.

If you total up the three instances above it only amounts to 6-7 hours. But you can do this constantly, every day. Before I start any task I ask myself: "Could one of my contractors possibly do this?"

Over the course of a month you can easily save 20-40 hours without much effort. These days I save 60-100 hours a month.

The roadblock that so many entrepreneurs encounter as they try to launch is thinking *they, or one of their co-founders, has to perform every task necessary to get their product out the door.*

Just for kicks I'm going to spit out a list of tasks needed to take a web-based product from idea to your first week after launch. Here we go:

- Niche Brainstorming & Mental Evaluation
- Niche Evaluation
- Niche Selection
- Product Selection
- Product Architecture
- Functional Design
- Database Design
- Graphic Design*
- HTML/CSS*
- UI Development (AJAX/JS)*
- Business Tier Development*
- Database Development*
- Creating Unit Tests*
- Creating UI Tests*
- Manual Testing*
- Fixing Post-Launch Bugs*

- User Documentation
- Installation Documentation
- Sales Website Site Map Creation
- Sales Website Copywriting*
- Sales Website Graphic Design*
- Sales Website HTML/CSS*
- Sales Website Programming*
- Sales Website Payment Integration*
- Product Delivery (via email, link on site, etc…)*
- Setting Up Email List
- Setting Up Domain Name & Web Hosting
- Setting Up Email Accounts & 800 Number
- Setting Up Analytics
- Pre-Launch Search Engine Optimization
- Pre-Launch Pay-Per-Click Set-up
- Initial Social Media /Viral Marketing*
- Pre-Launch Video Marketing
- Pre-Launch Partnerships
- Launch Press Release*
- Pre-Launch Email Marketing
- Pre-Launch Blogging or Podcasting
- And probably a few others…

There is a list of 37 tasks ranging in duration from 2 hours to a few hundred.

You'll notice many of them have asterisks next to them. These are the tasks that will be easiest to outsource – the tasks that require a technical or common skill that's not specific to your product.

Outsource your product architecture? Only for small applications.

Outsource your graphic design and HTML/CSS? Every time…

How to Avoid It

The bottom line is to start small, gain comfort with a contractor, and gradually increase the amount you outsource.

Outsourcing is a learned skill, and you're likely to screw it up your first time around. Start with non-critical tasks and be very specific in how they should be executed. At first it will seem like you could do the tasks faster than the time it takes to assign them, but as you get to know the person you're outsourcing to it will quickly begin to save you time. If it doesn't, then you need to look for a new resource.

Hiring a Virtual Assistant (VA) is a great way to get started with almost no financial commitment and a low hourly rate (around $6/hour overseas, $10-20/hour in the U.S.). We'll discuss VAs more in chapter 6.

Graphic design and HTML/CSS are also great ways to dive in. Graphic design is nice because it's not complicated and what you see is what you get. It's either good or it's not. Design is much easier to outsource than programming.

Finding decent designers is a little more challenging than finding a VA – Elance[9] is a good route, but asking around is even better.

Take a risk this month: outsource your first task and see where it takes you. When was the last time a single tool or work habit offered the opportunity to save 20-60 hours?

Changing Your Time Mindset

It's a big leap moving from employee to entrepreneur. One of the biggest adjustments is accepting that time is your most precious commodity.

[9] www.elance.com

Dollarizing
The phrase "dollarize" is used in sales to describe the approach of showing your prospect how your price is less expensive than your competition due to the amount of money they will save in the long run.

For example, you can dollarize a screw[10] by showing how your deliveries are always on-time, your defect rate is half that of your competitors and your screws can withstand an additional 500 lbs. of stress, each resulting in time saved in material handling and warranty calls.

If you take it a step further and you possess the appropriate data, you can approximate how much money your screws will save your prospect in a given year based on the number of times your competitors deliver late and how many defects the customer will avoid by using your screws.

It's a powerful technique and a way to turn an otherwise commodity purchase into a bottom-line savings.

Dollarizing Your Time
In the same vein, dollarizing your time is the idea of putting a theoretical dollar amount on each hour you work. If you value your time at $100/hour it makes certain decisions, such as outsourcing work to a $6/hour virtual assistant, a no-brainer.

Putting a value on your time is a foundational step in becoming an entrepreneur, and it's one many entrepreneurs never take. Skipping this step can result in late nights performing menial tasks you should be outsourcing, and an effective hourly rate slightly above minimum wage.

It never seems like a good idea to pay someone out of your own pocket for something you can do yourself…until you realize the economics of doing so.

---

[10] Editor's Note: They do this extensively in Nevada.

Approaches to Dollarizing Your Time
There are two approaches to dollarizing your time. Choose the one that makes the most sense for your situation.

Approach #1: Freelance Rates
If you are a freelance developer or consultant, you probably have an hourly rate. This is a good place to start. If you bill clients $60/hour, then an hour of your time is worth $60.

If you don't perform freelance work, do a search on Craigslist or Guru[11] for freelancers *in your local area* with similar skills. As a developer with a few years of experience you'll likely see rates in the $40 and up range. Frankly, if you have no other information, $50/hour is a good number to start with.

Approach #2: Salary
If you don't perform freelance work or have difficulty finding comparative freelancers online, another approach is to divide your current salary + benefits by 2,000 (the approximate number of hours worked in a year), rounded up to the nearest $5 increment.

It varies widely, but a typical benefits package including 401k matching, disability insurance, health care, and time off can range from 20-45% of your salary. You can come close to determining the real dollar amount using your pay stub and a bit of math, but if you just want to take a swing at it use 30%.

So if your salary is $60,000 per year, 30% of that is an additional $18,000 making your effective salary $78,000. $78,000 divided by 2,000 gives you an hourly rate of approximately $39/hour, or $40/hour when rounded up to the nearest $5 increment.

Be aware that freelance rates are nearly always higher than salaried rates because freelancers spend a portion of their time

[11] www.guru.com

on non-billable tasks such as invoicing, marketing, sales, etc… They have to increase their billable rate to make up for these non-billable hours.

Ultimately it's up to you, but I would tend towards using the higher freelance rate for your time, especially since it's closer to what you would receive on the open market if you chose to pursue freelance work.

Keep in Mind: Desired Earnings

Realizing your time is worth $50/hour is the first step; the next step is actually generating $50 for every hour you work, and the third step is figuring out how to make your time worth $75 or $100/hour. If you continue to think your time is worth $50/hour it will to stay at $50/hour.

$100/hour is a good long-term goal to shoot for. If you've done your research on one-person software companies (which are similar in economics to small software startups), the reality for most tends to be closer to $25/hour[12].

If you are making $25/hour as an entrepreneur you are doing something wrong. Improve your marketing, grow your sales, find a new niche, outsource and automate. $25/hour is not an acceptable dollarized rate for a startup.

While you won't be earning anywhere near $50/hour when you begin building your product, once you launch you should aim to hit that number within 6 months. In the early stages, your dollarized rate is a mental state but you want to make it a reality as soon as possible. Once you've succeeded, then you can work towards increasing it.

Realizations

Several realizations stem from dollarizing your time.

---

[12] http://blog.businessofsoftware.org/2007/09/start-a-softwar.html

Realization #1: Outsourcing is a Bargain
Once you've established you're worth $50/hour, paying someone $6/hour to handle administrative tasks or $15/hour to write code seems like a trip to the dollar store.

Outsourcing aspects of your business is the single most powerful approach I've seen to increasing your true hourly rate as an entrepreneur. If I didn't outsource my administrative tasks, my effective hourly rate would plummet.

Realization #2: Keep Work and Play Separate
Wasting time is bad. Boring movies, bad TV, and pointless web surfing are expensive propositions. If you aren't enjoying something, stop doing it.

I need to re-iterate here: I'm not saying you should never relax, have fun, watch movies, play with your kids, watch TV, or surf the web. I'm saying that you should be deliberate about your work and your free time to get the maximum benefit from both. In other words:

> *Work hard and play hard, but never do both at once.*

Numerous times throughout the day ask yourself:

> *At this very moment am I making progress towards crossing off a to-do, -or- am I relaxing and re-energizing?*

If I'm doing neither, evaluate the situation and change it.

If you aren't enjoying a movie, walk out.

If you're playing with your kids and working on your iPhone you're not really working or playing – you're doing both poorly. Put the iPhone away and focus on your kids; it will shock you how much more fun you have and how, after making this choice, you'll feel energized and ready to dive back into work.

The same goes for multi-tasking work in front of the TV. Your productivity level is around 50% when trying to do both. Most evenings you'll feel as if you worked the whole night but didn't get anything done.

Realization #3: Wasting Time is Bad

If your time is worth, say $75/hour, standing in line at the bank is painful. Sitting in traffic is another money waster – every non-productive, non-leisure minute you spend is another $1.25 down the drain.

Since it's not practical to assume you will never wait in line again, the best counter-attack is to have a notebook and pen handy at all times. Use this time for high-level thinking, something you may have a hard time doing in front of a computer.

It's amazing that we think we can remember our important thoughts. Due to the amount of information and chaos you consume each day, a thought stays in your head for a few seconds before it disappears. Perhaps you will think of it again, perhaps not. Writing down important ideas is critical to building a list of ways to improve your business.

With a notebook in hand, you'll find yourself having amazing insights while in line at the post office.

For years I've carried a notebook everywhere I go for this exact purpose. I use it to capture keyword ideas, product ideas, niche ideas, to-do's, and any other valuable information that surfaces.

Again, I'm not saying you should be working all the time – if you want to bring a magazine to read in line, by all means do it. If your mind needs to rest when you're running errands then use this as a time to re-energize so you can hit your work harder when you return to it.

The real statement here is that you should never find yourself *killing time*.

Realization #4: Information Consumption is Only Good When it Produces Something

The following discussion excludes consumption for pleasure, such as: reading a novel, watching *The Daily Show*, catching a movie, etc.

Consuming and synthesizing are very different things; it's easy to consume in mass quantity. It's much more difficult to synthesize information.

Have you ever read through an entire magazine only to realize you can't remember any specifics about what you just read?

As someone who likely enjoys consuming in large quantities, at some point you will realize that you are wasting an enormous amount of time. I highly recommend putting the following into

> *When reading blogs or books or listening to podcasts or audio books, take action notes.*

place:

Action notes are short- or long-term to-do items that apply directly to my businesses.

For example, I listen to several SEO podcasts. If they mention an interesting website, I make a note to check it out the next time I'm able.

As they mention a new SEO technique I create a specific to-do to try that approach on one of my websites. I make the action note specific so I can act on it quickly the next time I have a few spare minutes. If I were to write something general like "Google Webmaster Tools," it doesn't help me. But if I write "Create Google Webmaster Tools Account for DotNetInvoice," I can act on this quickly and cross it off my list without having to do much real thinking.

Action notes allow you to quickly determine which resources provide real value and which are fluff.

Since implementing action notes, I've canceled two magazine subscriptions, removed 40+ blogs from my RSS reader, and have become choosy about the audio books I buy.

This approach provides you with real-time feedback on the value of any consumable. A $4.99 audio book is actually a cost if it chews up 6 hours of your time and provides no actionable items.

Transitioning from Developer to Entrepreneur

You've likely realized that entrepreneurship and software development are two very different things. Software development is a subset of the skills an entrepreneur needs to launch and operate a successful startup.

If you've been writing code for years you've formed opinions and viewpoints that don't quite hold true in this new world. This lesson covers 9 realizations that will come to you at some point during your transition from developer to entrepreneur.

There's a lot of information here so don't feel as if you have to fully grasp everything today. If a few of them sink in and the

rest prepare you for what's to come, you will be in a better position to succeed.

Realization #1: Being a Good Technician is Not Enough
In *The E-Myth Revisited*[13], author Michael Gerber talks about the archetypes of running a business. They are: entrepreneur, manager and technician.

- The **entrepreneur** is the dreamer, the visionary, and the creative mind.
- The **manager** is the person who thinks about return on investment (ROI), near-term success, and productivity.
- The **technician** gets the work done. She follows the manager's guidance and is concerned about today's success.

95% of us are comfortable, and probably excel at, being technicians. This means you're good at writing code, producing something tangible, and cranking away on each task, moving one step closer to launch date.

But it takes more than a technician to run a successful business. It's critical to look ahead into the near-term and determine which features or marketing efforts will provide the best ROI (manager), and to think out a year or more to determine the long-term direction of your business (entrepreneur).

The first step is to determine your goals and objectives as we've been discussing in this chapter.

Without planning, organizing, systematizing, outsourcing, and marketing, all things you will shy away from as a technician, you will never make it past the $25/hour pit that many startups fall into.

---

[13] http://tinyurl.com/2bzsda6

Realization #2: Market Comes First, Marketing Second, Aesthetic Third, and Functionality a Distant Fourth
The product with a sizeable market and low competition wins even with bad marketing, a bad aesthetic, and poor functionality. Think QuickBooks in the early days, or any niche product you've ever seen that looked like it was written by a six year old but sold thousands of copies.

In the same market, the product with better marketing wins. Every time.

In the same market with equal marketing, the product with the better design aesthetic wins. Sure, a few people will dig deep enough to find that the "ugly" product has better or more functionality, but the product that wins is the one that has the best looking website and user interface.

Functionality, code quality, and documentation are all a distant fourth. I know that this sounds sacrilegious to a software developer, but unless you're marketing to software developers, your order of importance is market, marketing, aesthetic, function.

Realization #3: Things Will Never Be As Clear As You Want Them to Be
Writing code is cut and dry. There are different ways to accomplish the same thing, but in general you know how you want your application to behave and you just need to get it there. Your constraints are constant – the compiler behaves the same way it did the last time you compiled.

By comparison entrepreneurship, especially the marketing side, is never this clear. As we'll discuss in chapter 2, marketing is about math and human behavior. The math part is straight-forward. It's the human behavior that's going to throw you for a loop.

Even the foremost marketing experts in the world are not sure whether people will buy a new product. People with 20, 30 and

40 years of experience still have to take their best guess at what will succeed. They have to try things out and adjust as they go. They often do small roll-outs to test audiences and adjust the product or the message before unleashing it on the world.

You will have to do the same and it will involve a lot of guesswork at the start. That's a hard pill to swallow when you're used to making decisions based on fact. Instead, you have to take your best guess; then measure and tweak.

And then do it 20 more times until you succeed.

Realization #4: You Can't Specify Everything...But You Do Need a Plan

As developers, most of us have experience with the waterfall method of building software – write a detailed spec and build it as specified. With waterfall development changes are painful and time consuming.

You may be using an agile methodology these days, which is more in line with entrepreneurship than the waterfall approach. Define a long-term goal (launch your product), look at the next set of tasks that will get you one step closer to that goal, work, and re-evaluate in a week.

Before launch you may be able to get by using the waterfall method, but post-launch it will be a disaster. This goes for your development, support, and marketing efforts. As an entrepreneur, your #1 advantage is reaction time, and the waterfall approach reacts too slowly.

You can't specify everything, but you do need a plan to get you to your next release.

Realization #5: You Need to Fail Fast and Recover

If you haven't already, you will soon need to accept you are going to fail a lot. You will make bad decisions, waste time, waste money, run ineffective ads, miss deadlines, and release

buggy code. Each time this happens, you have to accept that you failed and move on.

The faster you fail and learn from your mistakes, the faster you will improve. Pretty soon your ads won't lose money, you'll get better at estimating level of effort, and you'll be sure to thoroughly test the complex parts of your code.

But you have to wade through that sea of failures before you can reach the other side. And this can be a hard thing to do.

Realization #6: You Will Never Be Done
Finishing a software product is a great feeling. The night you roll the new bits to the production server is indescribable. The feelings of relief, joy, and accomplishment are some of the most rewarding parts of developing software.

And you're never going to feel that way with your product.

Sure, you'll have releases and milestones. And you'll feel good the day you launch a new version.

But you will never feel "done." You will always have a list of features, marketing tests, potential partnerships, and new markets to take care of. And while the journey is itself a gift, never having the feeling of completion is something you need to get used to.

The idea of building an application and sitting back to collect a check is, unfortunately, a pipe dream. You have to continually invest in both your product and your marketing in order to remain successful.

Realization #7: Don't Expect Instant Gratification
The first month you launch you will be lucky to break $100 in revenue.

A product, marketing effort, and a reputation take time to build. But once they build they snowball such that the effort to

launch a new version of your product is miniscule by comparison, and your chances of success are much higher.

Once you have 5 releases under your belt, 1500 targeted visitors every month, a 500 prospect mailing list, and hundreds of incoming links…surprise! Things are easier. Much easier.

The effort of getting a new product off the ground is exponentially more than launching a new product once you have resources and experience behind you.

Don't expect that your work is over the day after you launch. That's the day work really begins.

Realization #8: Process is King
Documenting repeatable processes for anything you will do more than once is essential to your sanity.

It's true; you *can* fly by the seat of your pants and get by, but it makes you a hostage to your work.

If you've ever been a manager you probably like process and understand its benefits. If you're a developer you probably dislike process or see it as a necessary evil.

Startups, being lean and mean, seem like the perfect place to eliminate documents, have no systems, and no processes…but that's far from the truth.

Without process it's impossible to delegate, difficult to bring on a business partner, and easy to make mistakes. With processes in place it's much easier to sell your product if/when you want to make an exit.

The fact is, creating processes will bring you freedom through the ability to easily automate and outsource tasks. We'll discuss this in chapter 6.

Realization #9: Nothing about a Startup is a One-Time Effort

Many of us have the dream of launching our startup, investing time in the marketing effort, and from that day forward being able to focus on writing code. The problem is, nearly everything about a startup requires ongoing effort.

You have to invest time every month into marketing, development, support, SEO, AdWords, and every other aspect of your business. The dream of building an app that never breaks, never needs new features and possesses auto-pilot marketing are possible, but they will not come by accident.

To get to the point of an automated startup you have to choose your niche and your product wisely, and invest a large amount of time outsourcing and automating your business. Even then, support and feature development is the easiest part to outsource; marketing is one of the most difficult.

Conclusion

Realizing the differences between development and entrepreneurship is something that will benefit you in the long run as you pursue your dream of starting a company.

Understanding and embracing the entrepreneurial mindset will go a long way toward preparing you for the chaos that is a startup.

This concludes this sample of *Start Small, Stay Small*. If you are interested in reading the remaining six chapters I encourage you to visit www.StartupBook.net to purchase a copy of the book in PDF or paperback.