Secrets of a Software Startup

A Non-Tech Founders Guide to Assembling and Automating a Software Team

Nick Skelton

Secrets of a Software Startup

A Non-Tech Founder's Guide to Assembling and Automating a Software Startup

Nick Skelton

This book is for sale at http://leanpub.com/AnEntrepreneursGuideToHiringASoftwareTeam

This version was published on 2015-08-15

Contents

Who is this book for?

One of the most common questions people ask you when they meet you for the first time is:

"What do you do for a job?"

I bet you can think of the standard reactions to most of the common professions:

"Oh a Doctor wow I bet you make a heap of money!" "A lawyer well I actually had a parking ticket the other day that I don't want to pay..." "A photographer! Wow I hate having my photo taken." "You're in marketing.... Ahem... excuse me for a minute."

This is the most common response I receive:

"You write apps! I had a really good idea for an app! Tell me what you think about ..."

Usually it's a terrible idea, but ever so rarely it has potential. I really enjoy exploring interesting ideas with non-tech people because people who are completely naïve about the capabilities of technology have a childlike approach to what is possible. Every now and then, just like children, they will come up with something extremely insightful that makes you turn your head. The problem they face is making that idea come to life.

People rarely have good ideas. Even fewer will go out on a limb and spend some money making a prototype to see if this idea is both **possible** and **worthwhile**. Fewer still get positive answers to both of those questions. This book is intended for the ones who do. The ones who are at the stage of their startup where it's time to create their MVP[1] but have little or no experience in the field of *ongoing* software development. I stress the *ongoing* part because

[1]Minimum Viable Product

it's quite probable that if you are at the MVP or seed stage of your startup, you have created some kind of prototype or proof of concept. Perhaps you coded it up yourself (kudos!) or perhaps you got it done cheaply using a freelancer online. Let me be clear, this is not a guide for creating prototypes. Throwing together a proof of concept or a prototype is a "quick and dirty" approach to software development - which has it place and purpose. It got you this far. But when it's time to start creating the real product, its important to change your approach.

There are a number of paths that you can choose to take in order to get your product going. There is no right or wrong way. But if you are not able to write the code yourself – someone else will have to. And even if you can write the code yourself, you're going to have to work in a team eventually. Getting your product to market is not as simple as say choosing a dentist, or choosing a mechanic, its more like choosing an architect, a builder, a plumber, an electrician, a site manager and a landscaper, taking a few months off to watch them work. You want to be in control of the construction, it's your vision, you need to guide the ship, you need to be regularly (not constantly) monitoring its progress and correcting any digressions. This guide will help you understand the world of software just enough to allow you to effectively control the creation of your idea. It is not a technical book and it is not a book about creating apps, web sites or search engines. It's about how to get someone else to do that for you without losing control of your vision, or worse, smothering it and not letting it grow.

There will be sections of this guide that your peers will scoff at - advice that seems so counterintuitive and 'against the grain' that it will positively hurt your brain to accept it. Good. Being different is painful, but rewarding. There is no formula for success, but there are some basic guidelines and being different is right up there. Superficially plausible attitudes usually turn out the worst in the world of 1's and 0's so be careful about believing the things you hear while you're sipping a spritzer at the next-big-thing startup conference. This guide will educate you in the fields of software development and help you avoid the pitfalls that first time dev managers will fall into. It will help you understand the reasoning behind seemly counter intuitive advice and help you make a more informed high level technical decisions at critical junctures of your product's development that will benefit it's longevity. Even seemingly simple decisions can have unseen trade offs and it is important to, at worst, recognise after the fact so that you are able to adapt quickly and not continue to make bad decisions that will affect the quality and future of your product.

There are many myths about developing software and there are many levels and dimensions of experience. Software… it can be calculatingly ruthless and clinical, it can be art, it can be so exciting you stay awake into the small hours of the night and not notice, it can be so boring that you are struggling to stay awake at work at 3pm like a strung out junkie who just got his first hit in 4 days, you can be learning from a mad eastern european rocket scientist one day and teaching some punk kid the next, arguing about whether to use tabs or spaces for intentation formatting, then trying to sneak a pull request through without any unit tests the next. There are so many interesting personalities in software development, it like going to the zoo. I really love the line from Stephen Frear's movie 'Dirty Pretty Things': "You know what they say? Good at chess, bad at life." Some developers are brilliant at programming but hopeless at life. Some are experts in very specific areas and know absolutely zero about other areas. There are many different disciplines within software, but to the average person, its all just IT. A bit like when people assume that all lawyers are making great speeches in court rooms all day long, or make no distinction between a dentist and surgeon, or an architect and a builder. Hopefully this book will help you recognise the distinctions just in so far as you need and dispel the myths and assumptions that the layperson will make about software developers. In the end, by gaining an insight into their world you will better understand software development and thus be able to work more efficiently with them. And this can only give your idea it's best possible chance at success.

Part 1: Assemble

The first guide will help you choose your team. Think of any American action movie with a montage in the first half where the hero goes and assembles his team – a scientist, a thief, a joker, a mechanic, a weapons expert… This is what you have to do. You need to create your A-Team. Carefully construct the key personnel who will breathe life into your vision. I will focus on developers in this section because developers are really the core of software development. Other roles (designer and testers and eventually product owners) have become increasingly important, indeed essential, over the past 10 years so I will not leave them out. Still, developers are the ones who make up the bulk of any software team and are therefore the most important to focus on.

Part 2: Automate

The second guide will help you setup this team in an automated fashion so that you can effectively control and steer them in the right direction. Ideally you want to set them up like a production line, a well oiled machine with a small control box so that you can inspect the output and fiddle with the inputs so that the final product is exactly what you intend. Comparing a creative, professional and experienced group of people to a production line may sound cold, but every single developer I've ever worked with is most happy when they are working efficiently with clear goals, clear expectations, no waste, no interruptions and when progress is actually tangible. The source of all demotivation is being forced to spend time on something that is perceived as unproductive - politics, pointless meetings, timesheets, emergencies - to name a few. The second guide will outline how to effectively prevent or remove any such tasks.