THE MICROCONF READER

Startup Wisdom from Successful Founders

How to recognize your startup milestones, build a billion dollar business, restore customer trust, and run a software company on a few hours a week

Featuring selected essays from MicroConf 2011 speakers:

- ☑ **Sean Ellis of Freejit**
- ☑ **Hiten Shah of KISSMetrics**
- ☑ **Noah Kagan of AppSumo**
- ☑ **Rob Walling of Software By Rob**
- ☑ **Patrick McKenzie of Bingo Card Creator**
- ☑ **David Hauser of Grasshopper**
- ☑ **Mike Taber of Moon River Software**

Milestones to Startup Success | If It's Worth Doing, It's Worth Stealing | A Recipe for Building a Billion Dollar Business | Five Reasons You Haven't Launched | Running A Software Business On 5 Hours A Week | How to Break the Trust of Your Customers in Just One Day | How to Take on Goliath

DOWNLOAD THIS BOOK

This book is available for download at *http://bit.ly/MicroConf2*

ABOUT THIS BOOK

This book is a collection of essays by successful startup founders containing lessons learned launching and running their companies.

If you enjoy this eBook, you can see every one of these founders (plus several more) at MicroConf 2011, being held in Las Vegas on June 6th and 7th. Visit *www.MicroConf.com* to buy your ticket.

This book is free. We encourage you to email, forward and otherwise share it with the world.

Contents

Milestones to Startup Success

By Sean Ellis, *@seanellis*

Original URL: *http://startup-marketing.com*

Most entrepreneurs understand the importance of growth; the common mistake is trying to force growth prematurely. This is frustrating, expensive and unsustainable – killing many startups with otherwise strong potential.

Successful entrepreneurs usually have a good balance of execution intuition and luck. This was definitely the case at the two startups where I ran marketing from launch through NASDAQ IPO filings. While we didn't follow a specific methodology, our CEO was intuitive enough to know the right time to "hit the gas pedal" (as he called it). We didn't accelerate until verifying that the team had created a great product that met real customer needs and we could generate sufficient user revenue to support sustainable customer acquisition programs. It's taken years for me to realize that our growth was less a function of clever marketing tactics than beginning with something that customers truly needed. Some growth would have been automatic; the marketing team simply accelerated this growth.

Several startups later I have a much better understanding of the key milestones needed for a startup to reach its full growth potential. These are based more on observing universal truths than inventing some type of methodology. Reaching the full growth

potential of your startup requires focus, specifically focusing on what matters when it matters. In my blog post on *the startup growth pyramid* I talk about the high level milestones you must achieve in order to unlock sustainable growth. This essay looks at it on a more granular level with links to several resources that provide additional details.

Day 1: Validate Need for Minimum Viable Product (MVP)

Before any coding begins it is important to validate that the problem/need you are trying to solve actually exists, is worth solving, and the proposed minimum feature set solves it. This can best be achieved by meeting with the prospects most likely to need your solution. Surveying can also be helpful in getting the perspectives of a broader group of people.

Eric Ries offers more *details on the minimum viable product concept in this post/video.*

Where's the Love?

Vinod Khosla, one of the most successful Silicon Valley VCs in history, once suggested to me that startups should think of their early users as a flock of sheep. He explained, "The flock always finds the best grass."

For you this means you should start looking for a signal about who loves your product and why as soon as you release your MVP. Most products have at least a few people that truly consider it a must have. These people hold the keys to the kingdom. Learn everything you can about them including their specific use cases and demographic characteristics. Try to get more of these types of people.

A good place to start collecting this information is the survey I've made freely available on Survey.io (a KISSmetrics product). You can read more about this *product/market fit survey in this blog post*.

If you're lucky you'll be able to use this early signal to *find the product/market fit*.

Expose the Core Gratifying Experience

Before founding my current startup, I advisedseveral startups to help themuncover their core user perceived value. Once we understood it, we could exposethis value in messaging optimized for response and remove complexity that prevented first time users from experiencing it. Often this meant burying or even completely eliminating features that didn't relate to this gratifying experience.

Metrics

Obsessing over metrics can wait until after you've achieved product/market fit – then they are critical to your success. Dave McClure has a *great video on startup metrics that matter* (relevant part is at about minute 2:20).

Most of the tools out there provide way too many irrelevant metrics and miss the essential few. Both Dave McClure and I are advising KISSmetrics on a solution to this problem.

Start Charging

Another key step before growing your business is to implement a business model. The *ideal timing for implementing your business model is discussed in this blog post*.

I've often heard the argument that startups are focused on user growth and prefer to delay revenue in the short term. I believe the *fastest way to grow is with a business model and explain why in this blog post*.

Extreme Customer Support

Now that you have a business model in place, your first marketing expense should be to expand the customer support team. Anyone who cares enough about your solution to contact customer support is a great source of insight about your target market. Also, customer support will uncover issues that will help you grow faster without spending. And fixing these issues will make it much easier to grow when you do start spending, multiplying the impact of dollars spent.

If your customer support team is overwhelmed now, I don't recommend trying to grow until you address the issues driving most support calls. Once you've addressed these issues you'll have fewer barriers to adoption and will be able to grow without overwhelming customer support.

This will enable customer support to go above and beyond expectations, which is an important way to drive customer loyalty and enhance word of mouth. This approach pays more dividends today than ever before – as I explain in *this essay on Social Media*.

Brand Experience Over Brand Awareness

Back in the "Dotcom Bubble" days billions were wasted on brand awareness campaigns for startups. Today most entrepreneurs understand *that brand awareness campaigns are a waste of money* for startups.

Instead, it's much cheaper and more effective for startups to focus on creating a fantastic brand experience. While startups often realize the importance of brand experience, they focus on it too early, fine tuning things that customers don't care about. Instead, wait until you understand why certain customers love your product; then obsess over every element of this customer experience.

Apple is probably the best tech company out there on coordinating a perfect brand experience for its target users. I cover more on *brand experience in this blog post*.

Driving Growth

Once you've achieved all of the previous milestones, then you can focus on driving growth. CEOs must take an active role in driving customer growth whether or not they have an interest in marketing. Nearly all of the risk and upside in a startup is in your ability to gain customer traction and then drive scalable customer growth. The CEO should not abdicate this responsibility to the marketer.

It's important to stay aggressive and take all slack out of the market (make it completely uninteresting to pursue the market for any other competitor). Your early advantage

is the ability to iterate on the customer feedback loop and leverage strong customer loyalty to drive word of mouth.

While ROI lets you know if a user acquisition channel is sustainable, the key focus should be on exposing lots of the right people to your fantastic product experience. It's much easier to get passionate and creative about this than purely thinking about things from an ROI perspective. Of course positive ROI is essential for any customer acquisition program to remain in the mix.

When it's time to hire a marketing leader to partner with the CEO, *this post explains my recommendations for an ideal startup marketing leader*. The most effective startup marketers are relentless about experimenting with channels until finding things that work.

Start by building out free channels such as listing in directories and basic SEO. When you *begin building paid channels*, extra effort should be put into channels that show strong potential for scale.

Unfortunately you can't count on effective online tactics working forever. I've seen many hot online marketing tactics lose their effectiveness over time. This is because online tracking makes it easier for marketers to quickly figure out what actually works. As a result we start piling into the most effective tactics. Eventually *online tactics get saturated, as explained in this post*.

Business building

Fast growing businesses are difficult to manage. This is the point where you should bring in some experienced operations people if they aren't already on the team.

It Won't be Easy

Finally, the top three risks to growing via these milestones are:

- You lose patience and decide that one or more of the milestones really aren't that important.

- VCs and/or board of directors lose patience because you did not achieve conceptual agreement on this approach from beginning.
- You delude yourself into believing that for "our type of business" customers really don't need to consider our product a "must have". For us, "nice to have" is good enough.

Building a successful business is hard. Hopefully this milestone driven approach to growing your startup will make it a bit easier.

One other important note: it's hard to write an essay on "milestones to startup success" that covers every type of startup. Some startup types may need to reverse the order of some of these milestones. For example, with network effect businesses (EBay, social networks, eduFire, dating sites, etc.) user gratification increases with more users so there is a bit of chicken and egg here… Ad supported sites also benefit from earlier scaling.

If It's Worth Doing, It's Worth Stealing

By Hiten Shah, *@hnshah*

Original URL: *http://blog.kissmetrics.com/worth-stealing/*

How the Internet has created a culture of plagiarism, why it's a good thing—and the secret to copying and being copied without going under.

In September of 1993, version 1 of the program dubbed "the Internet's killer app" was released. It was called *Mosaic*, and its prodigy was Netscape Navigator.

A decade later, however, Netscape was long dead, and Internet Explorer had a death-grip on the browser market.

Today, Internet Explorer is struggling to keep abreast of Firefox, Chrome, and Safari.

How could Netscape go under with the Internet's killer app in its pocket? And how could a group of volunteers almost single-handedly tip the richest, most entrenched company in the PC market out of the browser-share top spot?

The Power Of Plagiarism

The electronic age has made it outrageously easy to copy other people's hard work. This is most obvious with the bits and bytes representing documents, images, and most controversially music and movies, which can be duplicated in seconds and cost only as much as the disk space they're stored on.

But it's not just created works that are being copied. The Internet has given a platform to every entrepreneur with a good idea and the gumption to test it. At the same time, it has given a platform to every entrepreneur *without* a good idea, but with the gumption to test somebody else's. So it was with browsers, and so it is with just about any other product, service, methodology, or business model you can imagine.

Which is exactly how we like it

With offline business models, you need bucketloads of green paper to break into most markets. That means less pressure on established businesses to compete and innovate and basically deliver products or services you want. When's the last time you felt good about a telco, for instance—an industry that's nearly impossible for newcomers to break into? But on the Internet, the openness and low cost of entry fosters innovation because anyone is free to take a crack at making money off an existing idea. So anyone who wants to profit from it—including whoever came up with it—has to do it the good old fashioned way: with ingenuity and hard work. That seems very much like the way it should be.

To be sure, software patents have tried to short-circuit this process by giving special privileges to whoever happens to patent an idea first. But the whole situation has turned into a long-range Mexican stand-off, with a lot of companies with a lot of patents training their guns on one another, none of them willing to open fire in earnest for fear of being gunned down themselves.

Fortunately, most ideas worth ripping off are either not yet patented, or unpatentable. So anyone with a good idea can try to make money off it—and anyone with a quick eye can try to beat them at their own game.

Some companies even have this philosophy written into their core values. *Zynga's "Innovate on best of breed mechanics"* directive is one example. And let's not forget that *Facebook, Twitter, Hulu and PayPal all got started by taking someone else's idea and doing it better*. Come to think of it, so did Google. And so did *KISSmetrics*.

Not if. When

How does this affect you? Simple—if your idea is worth copying, it *will* be copied. And if you see a good idea, you should probably copy it too. Good ideas are being disseminated more rapidly than ever through social media—and competition is more fierce than ever with entrepreneurs and startups looking to the web to save them from the recession. It's simply inevitable that you will be copied, and that you will copy in turn—sooner rather than later.

So, have you come up with a marketing strategy that's working well? Have you developed a business model that's raking in the dough? Have you invented a new product or service that is proving surprisingly successful? Someone is already thinking about copying it; and is probably even in the process of doing so right now. If they turn out to be really good at it, you could be in trouble.

Oh no—what to do?

Firstly, have a bit of a party—or at least take a moment to slap yourself on the back. Being copied means that someone thinks of you as a thought leader, an innovator, a guru, or whatever else floats your boat. You're already ahead because you got there first.

But secondly, realize that you might now be fighting for your life. Make the most of your lead by doing all the things that got you to the head of the pack to begin with. Don't be a Microsoft, who after claiming top spot in the browser market were then content to coast on their success—adding only the bare minimum of new features and standards support to new versions of IE despite these things being what made it so popular in the first place. Their complacency not only cracked open the door for Firefox's little toe, but actually fueled its development because people wanted more than IE offered.

You won't get far on inertia—although the bigger you are the farther you'll go. If you get complacent you'll be overtaken quickly. Even Firefox is starting to feel what it's like to have its heels nipped at, with Chrome becoming many people's browser of choice,

and Safari now available on Windows (who would have seen that coming a decade ago?)

Thirdly, look at *how* you're being copied. People usually innovate on the idea they're copying—and these innovations are often good. The Opera browser innovated on Netscape by adding tabs. Then Firefox copied that innovation because users liked it. Internet Explorer resisted, to its detriment, until IE7. And by the time Chrome came on the scene, tabs were just considered a natural part of a browser. You should take a similar approach, looking at how competitors are innovating on your ideas to get a sense for where you may want to improve your own offering.

But even that's not enough

Finally, even with a head start and a quick eye, it's a real gamble relying purely on your technical merits; on your expertise or on the superiority of your service, or whatever it is that people are copying. Because in the end, there's always a bigger fish, always someone smarter, always someone who can take your idea and just do it better, before you can catch up. Chrome is doing it to Firefox with its speed and minimalistic design. Will Firefox 4 change that? Maybe. But a lot of people have become Chrome *fans*, and won't easily switch back.

If you don't have something that turns your customers into fans, then you're sunk.

So what is this magic element? Thankfully it's the one thing that's almost impossible to copy—your ability to give your customers what they want.

This is your secret weapon, your ace in the hole. As long as you still deeply understand and care about your audience, you will have the edge over any newcomers who copy your ideas. So make sure you don't get so wrapped up in the "real work" you're doing that you forget about what sets you apart in a more fundamental way. Make sure you don't focus so hard on staying ahead that you forget about doing the one thing that can keep you from falling behind: knowing and caring about your customers more than anyone else.

A Recipe for Building a Billion Dollar Business

By Noah Kagan, *@noahkagan*

Original URL: *http://www.appsumo.com*

Ever since I was a little kid I wanted to be rich. I think most of us do, but never think of how to get there. I've spent a lot of time failing/ succeeding, so I thought I'd put together a delicious recipe that everyone can enjoy about how to build a billion dollar business. This recipe only serves one.

I once had a friend who was a child prodigy and HIGHLY respected in the Silicon Valley community. His only goal was to create a billion dollar business. Nothing else mattered. Ideas were either way too small or didn't monetize well enough. He took 1 year to build things and one grew very large. However, at the end of the year, he threw it all away. WHAT!?! What happened?

He had millions of users but no real product, no value, no love, no engagement, just tons of users with nothing for them. So what recipe / formula did he use? He put the business before the users.

A Recipe for Building a Billion Dollar Business

There is a simple recipe to creating a large large business:

- Focus. Focusing on how you can be a big business is the first mistake. Instead focus on creating something ultra valuable that people can't live with out. The question to ask is, "How would you feel if we weren't around anymore?" If they say they would die you've got a hit. Focusing on large billion dollar exits only sets your site way too far in the future and misses all the details you need to get there.
- Small. IBM, Facebook, Google, Mint, Microsoft, etc... All these companies were started by just a few people. They are not overnight billion dollar companies. They were a few people who came together to solve a problem. I doubt anyone planned from day 1 to make a billion dollars. Large things form from small. So don't fret your idea isn't worth a billion dollars today, just plant it like a seed in the garden.
- Time. Around 99.9999% of companies have not created over a billion dollars in value in less than 3 years. Are you in? Seriously, how long do you want to do it?
- Validate. Make sure before you start building something you have an idea of the problem you are solving. We spent 6 months building betarcade.com (it's not live, so don't bother checking). That's a lot of time and money we wasted, which could have been spent questioning users, doing surveys on surveymonkey and putting up ads to see what conversions would have been like for peoples interests. Ask for an upfront contract with people you talk to. Go even further and make them pay you ahead of time.
- Learn. The best thing you can do is fail. Not fail fast kind of stuff but just make tons of mistakes. You should be asking that after everything you do. Make sure not to spend all your time reading articles like this one, so you can balance it with creating, failing and learning from your mistakes.

- Organizational behavior. I am not the most organized person but here are

  a Few tips of successful companies:

  a) Daily check ins with everyone in the company about what they are doing that day.

  b) Have a complete list of all things you are working on that everyone can see

  c) Meetings are stand up and no more than 30 minutes. Anything else could be solved with less time / people

  d) No more than 5 people in a meeting.

- Objective. Work backwards. I had a friend who wanted to be a NY Times best seller. I asked how many sales did he need to make to the list. He had no idea. We then figured out the amount and worked backwards through marketing channels, priorities & strategies to guarantee that he would get the specific number of sales needed.. He did!
- In addition, here are some tools for being the best person to grow a company (ie. the appetizer)
  - tungle.me , use this for scheduling your meetings and stop wasting time.
  - rescuetime.com, see how you are spending your time on your computer.
  - macjournal. this blocks out your screen while you are writing so you can't be distracted
  - skype. turn off all instant messengers so you can focus on the work you are doing

- Market Size. This is possibly the most important of all. When Twitter was started it was a thing annoying tech guys did for their fanboys. Who would have thought where Twitter would be today. The thing to realize is the TAM (total available market) you have available or where your product is going as it could potentially grow to. For Twitter it would be looking at the newspaper or real-time advertising market.

And for dessert you need to consider the end users. Don't chase your tail around them. Build something you want for yourself. If you create something that you would actually pay money for, then it's hard to go wrong -even if you only end up with 1 customer.

Five Reasons You Haven't Launched

By Rob Walling, @robwalling

Original URL: http://www.softwarebyrob.com/2010/11/10/five-reasons-you-havent-launched/

This post is an accusation. A call to arms. A sharp stick that says "get off your ass and make something happen."

But I didn't write it for you; I wrote it for myself. Every one of these reasons has haunted me at one time or another over the past 10 years. Many a moon ago I thought I was the only person who struggled with them. Now I have several conversations a week that indicate otherwise.

These reasons will come to life every time you start something new, be it an application, a website, a book or a presentation. Excuses don't discriminate based on what you're creating.

So with that, here are five reasons you (and I) haven't launched…

Reason #1: You're Still Trying to Find the Right Idea

Give yourself a month. If you spend a month of "thinking" time, interspersed with a few hours a week researching ideas and you still haven't settled on one, close up shop.

Keep the day job. Hang around the water cooler. Become a lifer.

If you can't find a worthwhile niche in 30 days of intense thought and research there is trouble ahead, sir. This is an important decision, and yet it's just one on your path to launch. There are thousands more that need to be made before you'll get there.

If you can't make this decision in 30 days, save yourself the time and aggravation of trying to launch a startup.

The clock starts today.

Reason #2: You're Set on Doing Everything Yourself

Yep, I *am* going to say it again.

One of the most *time consuming startup roadblocks* is your need to control every detail and do every piece of work yourself. When you're scraping together 15 hours a week of night and weekend time, outsourcing 5 hours a week makes you 33% more productive.

You may be able to slice PSDs, but there *are people* who can do it faster and better than you. For $159 save yourself several hours of time.

You may be able to write web copy, but *there are people* who can do it faster and better than you.

Depending on your skillset, the same goes for graphic design, theme creation, creating unit tests, and a slew of other pre-launch tasks.

Time is one of your most precious commodities. Conserve it with a passion.

Reason #3: Hacker News, WoW, and [Insert Distraction Here]

Distractions are everywhere, including new-fangled distractions that disguise themselves as productive work (think Twitter).

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 and

evaluating something you may not need until 6 months down the road? And frankly, you'll probably never need it.

Another common distraction masquerading as productivity is reading business books. I've already *beaten this one to death* so I'm not going to re-hash it here. Suffice to say, most business books are the entrepreneur's Kryptonite.

Avoiding distractions takes discipline, and discipline is hard. Especially when building your product is supposed to be fun.

Wait, this is supposed to be fun, right?

I hate to be the bearer of bad news, but maybe building a product isn't quite what most blogs, books and magazines make it out to be. Maybe it's actually a long succession of hard work, late nights, and a boatload of discipline.

For some people that's fun. For others, not so much. You should decide which camp you're in before you get started.

Reason #4: You're Busy Adding Features (That No One Will Use)

When was the last time you spoke with someone who is planning to use your application in the wild?

Not your friend who's testing it out to make sure it doesn't crash when you login, but a real customer who entered their production data and told you they are anticipating the release of your application more than Iron Man 3?

If you've not working directly with at least two customers you have no idea if what you're building is adding value. Or a total waste of time.

Reason #5: You're Scared

We all are.

In my experience, fear is the #1 reason that keeps people from launching. Except most don't recognize it as fear because it manifests itself in other ways. Needing to make

everything perfect, to add one more feature, or to read one more marketing book are all ways that fear turns itself into excuses.

This is how fear really works. It keeps you from launching by tricking you into thinking you have real work to do, when that work is actually pointless busy work created to stave off your launch. Because launching is scary.

Actually, it's *terrifying*.

But we're all terrified at one time or another. You just have to deal with it and move on or you'll never get your product out the door.

Bonus Reason: You *Have* Launched, But No One Noticed

You had a great idea for a product. You went and built it in your bedroom. And now it's available for sale. Hooray! You made it to launch. Only problem: *no one noticed*.

You missed one minor detail in your mental business plan. You need a targeted, sustainable strategy for bringing prospects to your door. Or in this case, your website. If it's not both targeted *and* sustainable you are out of luck.

You can have a great launch day with a link from TechCrunch and an article on Mashable. But then it goes away. The traffic dies. 10,000 uniques the first month, 500 the second. 300 the third.

How can you build a business on 300 unique visitors a month?

The answer is: you probably can't. You need to figure out how targeted prospects are going to find you consistently, month after month. Because 10,000 unique visitors in a month is a nice bump in sales…nothing more.

What are your best bets? I know you want me to say "fun" and "sexy" things like Facebook and Twitter, but alas it's the un-sexy things like SEO, building an audience (blog or podcast), and having an engaged mailing list that work over the long-term.

(I have a lot to say about these un-sexy strategies. If you want more info check out my *startup book* for an entire chapter on this topic).

Running A Software Business On 5 Hours A Week

By Patrick McKenzie, *@patio11*

Original URL: *http://www.kalzumeus.com/2010/03/20/running-a-software-business-on-5-hours-a-week/*

Some four years ago, I started *Bingo Card Creator*, a business which sells software to teachers. At the time, my big goal for the future was eventually making perhaps $200 a month, so that I could buy more video games without feeling guilty about it. The business has been successful beyond my wildest expectations and has made it possible to quit my day job at the end of this month. The amount of time I've spent on it has fluctuated: the peak was the week I launched (50 hours in 8 days), a very busy week in the last few years spiked up to as many as 20 hours, and the average over the period is (to my best estimate) about 5 hours.

During the majority of the time I've had the business, I've also been a Japanese salaryman at a company in Nagoya. For those of you who are not acquainted with the salaryman lifestyle, I leave the office at 7:30 PM on a *very good day*, and have an hour and a half of commute both ways. In our periodic bouts of crunch time, such as the

last three months, I end up sleeping at a hotel next to the office (about 25 times this calendar year).

I'm not saying this to brag about my intestinal fortitude — this schedule is heck on your body and life, and absolutely no one should aspire to it. That said, I snort in the general direction of anyone saying a nine-to-five job is impossible to juggle with a business because "businesses require 100% concentration".

Here are practical, battle-tested ways for you to improve the efficiency of your business and deal with some of the niggles of partial self-employment. They'll hopefully be of use whether you intend to try running it in your spare time or just want to squeeze more results out of the time you're already spending. Many of these suggestions are specific to the contours of running a software business on the Internet, which has a lot to recommend it as far as part-time businesses go — take care before trying these willy-nilly with an unrelated industry. (Part-time pacemaker research is probably not the best idea in the world.)

Time as Asset; Time as Debt

The key resource if you're running a business by yourself is your time. Other businesses might worry about money — however, you've probably got all your needs and then some covered by your day job salary, and capital expenditures in our business are so low as to be insulting. (I started my business with $60. Literally.) And the key insight about time is that software lets us take the old saying about how "Everyone gets the same 24 hours per day" and break it open like a pinata.

Time *can* be stored. One of the great features about currency is that it functions as a store of value: you create some sort of value for someone via your labor, trade that value for currency, and then the currency will retain value even after the physical effect of the labor has faded. For example, a pumpkin farmer might not be able to conveniently store pumpkins, but if he sells them the currency will (under normal circumstances) not rot.

Most people think, intuitively, time *always* rots. You get 24 hours today. Use them or lose them. The foundation of most time management advice is about squeezing more and more out of your allotted 24 hours, which has sharply diminishing returns. Other self-help books exhort you to spend more and more of your 24 hours on the business, which has severely negative effects on the rest of your life (trust the Japanese salaryman!)

Instead of doing either of these, **build time assets**: things which will save you time in the future. Code that actually does something useful is a very simple time asset for programmers to understand: you write it once today, then you can execute it tomorrow and every other day, saving you the effort of doing manually whatever it was the code does. Code is far from the only time asset, though: systems and processes for doing your work more efficiently, marketing which scales disproportionate to your time, documentation which answers customers' questions before they ask you, all of these things are assets.

The inverse of time assets is **time debt**. Most programmers are familiar with technical debt, where poor technical decisions made earlier cause an eventual reckoning where forward progress on the program becomes impossible until the code is rearchitectured. Technical debt is one programmer-specific form of time debt. Basically, time debt is anything that you do which will **commit you to doing unavoidable work** in the future.

Releasing shoddy software, for example, commits you to having to deal with customer complaints about it later. So don't do that. Better yet, rather than a useless bromide like "don't release bad software", spend time creating systems and processes which raise the quality of your software — for example, write unit tests so that regressions don't cause bugs for customers.

However, **not all time debt comes from intrinsically negative activities**: there are many things that successful businesses do which cause time debt and *you probably do not have the luxury of engaging in them*. For example, high touch sales processes incur time debt almost as soon as you put out your shingle: you're committed to spending many, many hours wining and dining clients, often on a schedule that you

cannot conveniently control. That is generally a poor state of affairs to be in for a part-time entrepreneur, even though there are many wonderful businesses, small and large, created in high-touch industries.

Code Is About 10% Of Your Business. Maybe Less.

Are you considering starting up a business because you wish to work on wonderfully interesting technical problems all of the time? **Stop now** — Google is hiring, go get a job with them. 90% of the results of your business, and somewhere around 90% of the effort, are caused by non-coding activities: dealing with pre-sales inquiries, marketing, SEO, marketing, customer support, marketing, website copywriting, marketing, etc.

Bingo Card Creator has been memorably described as "Hello World attached to a random number generator." If anything, that probably overstates its complexity. Customers do not care, though — they have problems and seek solutions, regardless of whether the solution required thousands of man years of talented engineers (Excel) or one guy working part-time for a week. (You'll note that you can make bingo cards in Excel, too. Well, you could. Many people can't. If I sell to them, I don't necessarily have to sell to you.)

Relentlessly Cut Scope

37Signals had many good ideas in their book *Getting Real*, but probably the best one is to "Build Less". **Every line of code you write is time debt**: it is another line that has to be debugged, another line that has to be supported, another line that may require a rewrite later, another line that might cause an interaction with a later feature, another line to write documentation for.

Cutting your feature set to the bone is the single best advice I can give you which will get you to actually launching. Many developers, including myself, nurse visions of eventually releasing an application… but always shelve projects before they reach completion. First, understand that software is a work in progress at almost every stage of maturity. There is no magic "completion" day on an engineer's schedule: "complete" is 100% a marketing decision that the software as it exists is Good Enough. If you have

to cut scope by 50% to get the software out the door, you're not launching with a 50% product: you're launching with 100% of the feature set that is implemented, with 100% of (hopefully decent) ideas for expansion in the future.

Pick Your Problem Well

Long before you sit down to write code, you should know what your strengths are and what your constraints are. If you can only afford to spend 10 hours a week and your schedule is inflexible, then anything which requires calling customers in the middle of the day is out. Scratch B2B sales for you. If you have the graphical skills of a molerat, like myself, you probably should not develop for iPhones. (Minor heresy: while Mac developers are very graphically intensive people who will buy software just to lick it if the UI is good enough, many Mac users are just regular people. My Mac version has a conversion rate fully twice that of the Window version, and it is not noticeably pretty.)

Some people profess difficulty at finding applications to write. I have never understood this:*talk to people*. People have problems — lots of problems, more than you could enumerate in a hundred lifetimes. Talk to a carpenter, ask him what about carpentry sucks. Talk to the receptionist at your dentist's office — ask her what about her job sucks. Talk to a teacher — ask her what she spends time that she thinks adds the least value to her day. (I'll bet you the answer is "Prep!" or "Paperwork!")

After you've heard problems, find one which is amenable to resolution by software and that people will pay money for solving. One quick test is to see whether they pay money for solving the problem currently: if people are spending hundreds of thousands of dollars on inefficient, semi-manual ways to do something that you could do with Hello World and a random number generator, you may be on to something. (For example, if you knew *nothing* about the educational market, you can infer that there are at least several hundred thousand dollars sold of reading vocabulary bingo cards every year, just by seeing those cards stocked in educational stores across the country and doing some quick retail math. So clearly people are spending money on reading vocabulary bingo. It isn't *that* much a reach to assume they might pay money for software.)

Other things you would look for in your idea are anything you see yourself using in your Benefits section of the website to entice people to buy it. (Benefits, *not* Features. People don't buy software because of what it does, they buy it for the positive change it will make in the life.) If you think "People should buy this because it will make them money, save them time, and get them back to their kids faster", then you probably have a viable idea.

Another thing I'd look for prior to committing to building anything is a marketing hook — something you can take advantage of to market your product in a time-effective way. For bingo cards, I knew there were more activities possible than any one company could ever publish, and that gave me hope that I could eventually out-niche the rest of the market. (This is core idea still drives most of my marketing, four years later.) Maybe your idea has built-in virality (nice if you can get it — I really envy the Facebook crowd sometimes, although I suppose they probably envy having a customer base which pays money for software), a built-in hook for getting links, or something similar. If you can't come up with anything, fix that before you build it.

This should go without saying, but talk to your customers prior to building anything. People *love* talking about their problems to anyone who will listen to them. Often they won't have the first clue about what a solution looks like, but at the very least repeated similar emotional reactions from many people in a market should tell you that the problem is there and real. After that, it is "just" a matter of marketing.

One note about business longevity: you will likely be involved in this business until you decide to quit. That means planning for the long term. Markets which change quickly or where products rot, such as applications for the iPhone (which have a sales window measured in weeks for all but the most popular apps) or games (which have constantly increasing asset quality expectations and strong fad-seeking in mechanics/themes/etc) interact very poorly with the constraints you are under. I would advise going into those markets only with the utmost caution.

Get Your Day Job Onboard

Don't do work on your business at your day job. DO NOT do work on your business at your day job. Do NOT do work on your business at your day job. It is morally and professionally inappropriate, it exposes you to legal liability (particularly if your business ends up successful), and it just causes headaches for all concerned.

As long as you follow that one iron law of doing a part time business, all other obstacles are tractable. Many engineers these days code outside the clock — for example, contributing to OSS projects. Tell your boss that you have a hobby which involves programming, that it will not affect your performance at work, and that you want to avoid any misunderstandings about who owns the IP. You can do something culturally appropriate to actually effect that: it might involve a contract, a memorandum of understanding, or even just a promise that there is no problem.

(Aside: I know many Americans consider the last option shockingly irresponsible. My ability to prevail over my employer — a major multinational — in a lawsuit is effectively nil. A contract is just a formalization of a promise. In Japan, the ongoing relationship with my bosses is the part of the agreement that provides security, not the piece of paper.)

One sweetener you can offer any employer: providing you with discretion to continue with your hobby costs the employer nothing, but it will result in you getting practical experience in technologies and techniques you wouldn't normally get at the day job, and they can then make use of that expertise without having to send you to expensive training or seminars. I generated conservatively six figures in business for my day job as a result of things I learned from my "wee little hobby." Feel free to promise them the moon on that score — all they have to do in return is not object to your hobby.

Speaking of day jobs: if you know entrepreneurship is in your future, you might pick a job which dovetails nicely with it. Prior to becoming a salaryman I was employed by a local government agency which had stable salaries and a work-day which ended at 4:30 PM. Hindsight is 20/20, but that would have been perfect for nuturing a small business

on the side. (What did I do with my free time back when I had so much of it? I played World of Warcraft. *Sigh* Youth, wasted on the young…)

Avoid Setting Publicly Visible Deadlines

One thing I did not know four years ago was how dangerous it is to promise things to customers. For example, suppose a customer asks for a feature which is on the release roadmap. I might, stupidly, commit to the customer that "Yes, this will be available in the next release, which I hope to have ready on next Monday." If the day job then has me spend the rest of the week at the hotel, or I have a family emergency, I will miss that deadline and have one ticked-off customer to deal with. That is 100% avoidable if you simply *don't commit to schedules*. (Also note that committing to a schedule is time debt, by definition. If you ever say "Yes, I will implement that", you've lost the ability to decide not to implement it if your priorities change.)

One of the most useful things I learned in college was a line from my software engineering professor. "The only acceptable response to a feature request is: 'Thank you for your feedback. I will take it under advisement and consider it for inclusion in a later version of the software.'" That line actually works. (There are industries and relationships in which it won't work — for example, if you're in a regulated industry and the regulations change, you can't fob the regulatory authority off with that. *Don't be in a regulated industry*.)

Release schedules are not the only type of deadline out there. Ongoing relationships with freelancers will occasionally have deadline-like characteristics, too. For example, if you have a pipeline where you generate requests for work and then the freelancer fills it, if you unexpectedly are unable to do your part, the freelancer will be idle. Thus, you want a bit of scheduling flexibility with them, a store of To Be Done On A Rainy Day requests queued up, or a rethink of your relationship such that *your* brain is not required for them to be able to do *their* job.

Cultivate Relationships With Effective Freelancers

Dealing with outside talent is one of the most important skills of being a part-time entrepreneur. It lets you work more hours than you have personally available, it lets you use skills that you don't possess, and especially *when combined with software you've written* you can do truly tremendous things with with a little bit of elbow grease. Many folks get started with freelancing from posting to sites like Rentacoder (awesome article about which *here*) or Craigslist. That is fine — everyone has to start somewhere. However, you'll quickly find that there is literally a world of people out there who are willing to work for $1.50 an hour… and would be terrifically overpaid at that price.

My suggestion is that, when you find a freelancer who you click with, hold onto them for dear life. Pay them whatever it takes to keep them happy. Additionally, since most clients are just as incompetent as most freelancers, don't be one of the flakes.

- Pay freelancers as agreed, promptly. I jokingly refer to my payment terms as Net 30 (Minutes), and that ends up being true 90% of the time.
- Provide sufficient direction to complete the task without being overbearing. (Freelancers with a bit of personal initiative are worth their weight in gold.)
- Don't schedule things such that freelancers are ever blocking on you or that you are ever blocking on freelancers. You have all the time in the world if you get things done well in advance of need. For example, I just got my St. Patrick's Day wordpress theme done — for *next* year. If I was getting the *Easter bingo* site cranked out now, any hiccup would mean it missed my window. (Technically speaking it would already be too late for SEO purposes, but that is a long discussion.)
- Recurring tasks are a great thing to systemize and outsource. You can write software to do the painful or boring bits, greatly increasing productivity, and as your freelancers get more experienced at the task you take on less time debt for explanation and review of their work.

Speaking of which, the most successful freelancing relationships are ones where you correct the labor market's estimation of someone's value. (That is the positive way to say "You spend much less on them than you'd pay someone else for the same work *and* they're happy to get it because you're the highest paying offer.") Much ink has been spilled about how the globalization of labor makes it possible to get work done by folks in low-wage countries. To the extent that you identify skilled, reliable workers, this is certainly one way to do things, but it is not the only way. The current economic malaise has left many folks in high-wage countries either unemployed or underemployed. In addition, the labor markets have huge structural impediments to correctly valuing the expertise of stay-at-home mothers, retirees, and college students. All of those are potentially good resources for you.

Understand the Two Types of Time

There are two types of time involved in business: wall clock time and calendar time.

Wall clock time: minutes/hours which you spend actually working.

Calendar time: days/weeks/months/years where time passes so that something can happen.

We expect the world to be very, very fast, because the Internet is very, very fast, but when dealing with non-Internet processes we are frequently reminded of how slow things are.

Paul Graham *mentions this* as one of the hard things to learn about startups. I really like his metaphor for how to deal with it: fork a process to deal with it, then get back to whatever you were doing. For example, while Google rebuilds its index in a matter of minutes these days (this blog post will be indexed within fifteen minutes of me hitting the post button, guaranteed), getting a new site to decent rankings still takes months of calendar time. That doesn't mean you stand around waiting for months — you get your site out and aging as fast as humanly possible, and then start working on other things. **Get good at task switching** — you'll be doing it a lot. (I literally just alt-tabbed to Gmail and squashed a support inquiry.)

You can incorporate calendar time into your planning, too, and since it is essentially free to you (you're planning on being here in a week, right?) it is often advantageous to do it. For example, A/B testing requires lots of calendar time but very little wall-clock time: you spend 15 minutes coding up the test and then have to wait a week or two for results. That works very, very well in a part-time business. Often, you can get into a rhythm for feedback loops like that. Do whatever works for you: for me, Saturday typically sees me end my old tests and start new ones.

Avoid Events, Plan For Processes

There is a temptation to see business as series of disconnected events, but that should probably be avoided. For example, you might see a dozen emails as a dozen emails, but it is probably just as true that it is six of Email A, 3 of Email B, and three emails with fairly unique issues. You should probably turn your response to Email A and Email B into some sort of process — address the underlying issue, write your web page copy better, add it to your FAQ, create an auto-text to answer the problem, etc etc.

Similarly, spending your time on things which help your business once is almost always less effective than making improvements which you can keep. For example, running a sale may boost sales in the short term, but eventually the sale will end and then you cease getting additional advantage from it. There is a time overhead assorted with running the sale: you have to promote it, create the graphics, code the logic, support customers who missed the sale by 30 minutes but want the price (give it to them, of course), etc etc. Spend your time on building processes and assets which you get to keep.

Another example: attempting to woo a large blog to post about you may require quite a bit of time in return for one fleeting exposure to a fickle audience. Instead, spend the time creating a repeatable process for contacting smaller blogs, for example something along the lines of *Balsamiq's very impressive approach*. (Other examples: repeatable piece of linkbait such as the *OK Cupid's series on dating* also works, or a repeatable method of building linkable content, or a repeatable way of convincing customers to tell their friends about you.)

You can also avoid spending hours on incident response if you spend minutes planning your testing and QA procedures to avoid it. When they fail — and they *will fail* — fix the process which permitted the failure to happen, in addition to just responding to the failure.

Document. Everything.

I'm indebted to my day job for teaching me the importance of proper internal documentation. As weeks stretch into months stretch into years, no matter how good of a memory you have, you will eventually have things fall through the cracks. Your business is going to produce:

- Commit notes. Thousands of them.
- Bug reports.
- Feature requests.
- Pre-sales inquiries
- Strategic decisions
- Statistical analyses

… etc, etc. The exact method for recording these doesn't matter — what matters is that you will be able to quickly recall necessary information when you need it.

I tend to have short-term storage and long-term storage. Short term things, like "What do I need to do this week?", get written down in a notebook that I carry with me at all times. (I lock it in the drawer when I get to work, but feel no compunction about sketching things on my train ride.) Things that actually need to get preserved for later reference go into something with a search box. This blog actually serves as a major portion of my memory, particularly for strategic direction, but I also have SVN logs (with obsessive-compulsive commit notes… often referencing bugs or A/B tests by number), email archives, and the like. (One habit I picked up at the day job is sending an email when I make a major decision outlining it and asking for feedback. Note this works just as well even if you're the only person you send it to — at least you'll force

yourself to verbalize your rationalizations and you can compare your expectations with the results later.)

There are a million-and-one pieces of software that will assist in doing this. My day job uses Trac, which has nice SVN integration. I have heard good things about 37Signals' stuff for project planning/management and also about Fogbugz for bug tracking. Use whatever works for you.

Note that quality documentation of processes both prevents operator error and makes it possible for you to delegate the process to someone else. Also, if you have eventual designs on selling this business, comprehensible and comprehensive documentation is going to be a pre-requisite.

Dealing With The Government

I've been pleasantly surprised how little pain I've suffered in dealing with the government. Part of this is because software is such a new industry that we often slide by on regulation — if I ran an actual Italian restaurant instead of the software analogue, I would have to keep health inspectors happy on a regular basis, but there is (thankfully) no one auditing my code quality. Speak with competent legal advice if you're not sure, but for the most part the only thing Japan and America want from me is that I pay my taxes on time.

Paying taxes is weeks of hard work really freaking easy. The typical Italian restaurant has to do lots of bookkeeping involving thousands of sales, most of them involving cash, juggle record-keeping demands for half-dozen employees, and has expenditures ranging from rent to wages to capital improvements to food with a thousand rules about depreciation, etc, to worry about. By comparison, the typical software business gets half of bookkeeping *for free* (if you can't tell me to a penny how much your software business has sold this year with a single SQL query… well, I don't know whether to deride your intelligence or congratulate you on your evident success), we have absurdly high margins so if you forget to expense a few things it won't kill you, the number of suppliers we deal with is typically much lower, and the vast majority of what we do is amenable to simple cash accounting.

Additionally, your local government almost certainly has a bureau devoted to promoting small businesses. They are happy to give you pamphlets explaining your legal responsibilities — in fact, sometimes it seems the only thing they do is create ten thousand varieties of pamphlets. Your local tax office will also fall over backwards telling you how quick and easy it is to pay them more money.

Incorporation? Incorporate when you have a good reason to. (I still don't, but I might do it after I go full-time, largely for purposes of dealing with Immigration.) If you're selling B2C software, your number one defense against getting sued is promptly refunding any customer who complains, and that pretty decisively moots the LLC's (oft-exaggerated) ability to limit your personal liability. You'll be personally liable for debts from the business, but since the business is fundable out of your personal petty cash that isn't the worst thing in the world. If sales collapsed tomorrow I'd be on the hook for my credit card bill, which runs about $1,200 a month — not a financial catastrophe for an employed professional, particularly when the business generates far more than that in profits well in advance of the bill being due. Sole proprietorship — i.e. merely declaring "I have a business" — is the most common form of business organization, by far.

Ask Someone Else About Health Insurance

I'm only putting this here to mention I have no useful information, because I live in a country with national insurance. That isn't a veiled political statement — I am not really emotionally attached to either model, I just don't have useful experience here. (My impression is that young single businessmen around my age are probably well-served with getting cheap catastrophic coverage.)

Keep A Routine, When Appropriate

Through sickness, health, and mind-numbing tedium, I've woken up every day for the last four years, checked email, gone through the day, checked email, and gone to sleep. This is the single best guarantee that I would deliver on the promised level of service to customers — almost all questions answered within 24 hours. There have been many, many weeks where this is literally all I've done for the business.

I try to keep creative work — such as writing, coding, or thinking up new tacts for marketing — to a bit of a routine, too, with flexibility to account for days where I'm not mentally capable of pushing forward. For example, generally I do planning for the week at dinner on Monday and have four hour block to the business on Saturday. If on Saturday it turns out that I can't make forward progress on the business, I clock out and go enjoy life.

Routines aren't limited to the business, either. They help me incorporate my other priorities — family, friends, church, gym, hobbies — into a schedule that would otherwise descend into total anarchy. (If you want to see what happens to the things that I don't prioritize when the day job starts knocking, well, suffice it to say that I was cleaning today and removed 13 pizza boxes from my kitchen table. I hope to put both cleaning and cooking back in the rotation after separating from the day job.)

Seek The Advice Of Folks You Trust. Disregard Some Of It.

One of the major things which pushed me to (a small measure) of success these last four years has been advice from the communities at the *Business of Software* boards and *Hacker News* and the writings of folks like Joel Spolsky, Paul Graham, and the 37Signals team. Much of the advice I received has been invaluable. I disagree quite strongly with some of it. When reading advice from me or anyone else, keep in mind that it is a product of particular circumstances and may not be appropriate for your business. And always, always, always trust the data over me if the data says I'm wrong. (That's the easy part. The hard part is trusting the data when it is overruling *you*.)

How to Break the Trust of Your Customers in Just One Day

By David Hauser, *@dh*

Original URL: *http://davidhauser.com/post/1306089659/how-to-break-the-trust-of-your-customers-in-just-one*

As entrepreneurs and human beings, we make mistakes everyday in our business, but most of the time the mistakes are small enough so that it doesn't land us in the press for all the wrong reasons. On Monday, October 11th, Chargify made a massive mistake in making the announcement about a change in pricing; as far as mistakes go, this one was pretty gargantuan. Plain and simple, we did a horrible job of communicating a change that wasn't driven by greed or stupidity, but was part of our desire to better support our customers that are really trying to grow viable businesses. I could sit here all day and try to convince you that our change in pricing really was part of a plan to provide better service, but really, does it matter? When it comes to matters like these,

the intentions do not matter as much as perceptions, emotions and how we treated the community.

Having spent the last two days focused on what we did wrong, and responding to the numerous inquiries and complaints from customers, I've had a great opportunity to identify where we screwed up. If that reflection were not enough, I also watched as two (1,2) posts about the Chargify price change made it to the top of Hacker News with over 180 comments (mostly negative). Then, of course, there was the article onTechCrunch, and a blog post on Inc. pointing out our blunder, too. To put it simply, the last twenty-four hours have sucked, and they should never have happened, but they did, and now we need to learn something and try to earn back our customers' trust.

Communicate early and often

Just one of the huge mistakes we made was sending one email, without any warning, notifying all customers of a large price increase. This broke a trust that we had developed with our customers over a long period of time, and will take much to repair. We should have communicated our need and desire to remove free plans and provided more information about how this would happen, and over a period of time leading up to the change. Maybe this could have been as simple as starting the communication three months ago, or as difficult as calling every single customer and talking to them on the phone to notify them of the change. Bottom line is that we should've done it better. It just wasn't right.

Perceptions matter, so be open with information

As a result of our horrible communication, the perception of the pricing change was that it was because we just wanted more money, but that wasn't the case. We should have shared the data we collected for over a year that demonstrated quite clearly to us that only *0.9%* of customers were paying us at all, and that there was a direct correlation between those that did not pay anything and a high volume of support requests. Even though this informed our decision to make a change in Chargify pricing, we didn't bother to share that with you. Mistakes like these are important lessons and

we learned that we should've told you a long time ago what we ourselves were finding in the data collected.

Free customers go out of business or never launch

Many of you have asked why we don't just go back to offering some kind of free service. A year ago, we here at Chargify thought it made sense to offer a free plan—after all, wouldn't it just allow more people to start and grow their business to the point where they'd be paying customers anyway? It's a good idea, but as it turns out, that's not what happens. In the past year, we discovered that those businesses that we thought would initially pay nothing and then grow into paying customers just never ever did; for the most part, launches never happened and they went out of business. The hard truth is that many, many people try to turn hobbies into businesses and it just doesn't work. While everyone deserves a shot to start a business, our theory that non-paying customers would eventually turn into paying ones just didn't pass the test when it was put into practice. Although we should've shared this with you so that our decision didn't seem out of the blue, we simply can't support free accounts and provide the kind of service we want, plain and simple.

"What bleeds, leads"

Everyone knows this phrase from our media-obsessed culture, and it holds true for tech blogs and the related community when a company makes a gargantuan error. Chargify's price change hit the top of Hacker News twice, garnered almost 200 comments, and then the icing on the cake for a shitty day were a couple of articles on TechCrunch and Inc.com. Like other web app startups, we had tried to get TechCrunch coverage for a long time, and although they loved covering one of our competitors, they never covered Chargify until it was time to report something negative. We can argue about the merits of even wanting to be on TechCrunch, as my friend Paras Chopra did in his post, "Demystifying the TechCrunch Effect," but we did actually get the coverage we wanted, and had very high signup days. So, while we should issue a big thank you to TechCrunch for the press, we don't plan to screw up as royally as this again, which

means we probably won't ever be covered by them in the future. That's fine; we'd rather have happy customers instead.

Free customers have the time to complain

There is a big difference between bootstrapping a business, which I have done a number of times, and trying to test a hobby to see if it is viable as a business.

Over the past year, we discovered that the customer that never paid had the highest support load. Once we made the announcement about the price change, the same applied to complaining about Chargify across multiple public channels. Those customers that were working on a hobby business, or just something they were not investing in significantly, seemed to have the time to tweet all day long, post multiple negative comments on every possible channel available, and shout the loudest. This is not to say we did not get valid complaints from great customers—boy, did we ever—but their complaints were well considered, included real information and were mostly in private forums.

Freemium gets a lot of talk; thankfully few use it

Everyone's always talking about freemium, but very few people actually use it, and we discovered this in looking at our customers for the past year. The reality was that less than *0.4%* of customers had any sizeable number of free customers on their accounts. For the small amount who did, Chargify has taken care of them and will not charge them. We should have communicated why the pricing was simplified to include just customers with no distention and handled the few edge cases better. Freemium is a hot topic these days, but far less people are actually using it than is widely reported.

Stand firm, but listen

Making a big decision that may change the direction of a business is not easy and you must stand firm in that decision, but be open to listening to and engaging the community. We will not offer a totally free option as that is not good for our business or for our

customers, but we did make a $39 plan available to those that supported us during our growth. Should we have offered this option before our major announcement? Yes. Would we love to go back and do it over again? Yes, but the reality is that's not happening, so we need to do the next best thing and support those who have supported us with a $39 plan.

Grandfathering

This is a huge and unsettled topic about which we are still getting feedback on each day. Maybe we should have offered a grandfathering option, maybe we should have given a grandfathering option to those that already integrated, maybe we will do all of this, but at this point we have not. The issue here is that it really depends on the business, the pricing change and how dramatic it is for each customer. Looking back, the best option would have been a grandfathering option which allowed the previous pricing but only included community support. Still an open topic.

After personally replying to more than one hundred tickets, tons of comments across multiple publications and on Twitter, threads on Hacker News and many other channels, it was important to look back on all of this feedback and see what went wrong so our team (and others) can learn from it. Regardless, we broke your trust, trust that took a massive amounts of time to build, and now we may never get it back. I don't have any neat solutions for you and I don't want to feed you crap. All I can say is that we have learned more from this mistake than from anything before and will use that knowledge to change the way we think about everything related to Chargify. That might not seem like enough of a mea culpa from me, but our desire has always been to empower entrepreneurs to succeed with real tools and solutions for growing business, and that mission hasn't changed. That was the driving force behind this pricing change, and whether you believe that or not is your call; we will show you our commitment to you, our customers, with our actions now.

How to Take on Goliath

By Mike Taber, *@singlefounder*

Original URL: *http://www.singlefounder.com*

One of the pieces of advice that I've heard doled out. Over the years is this. "Don't build a product that goes head to head against a company like X.", where X is usually Microsoft, Oracle, Google or some other large, public company with billions of dollars sitting on pallets in a dark bunker somewhere. Increasingly, the names mentioned are companies who are much smaller, but tend to have extremely large networks of users, such as Facebook or Twitter.

The reason these upstart companies have become so "dangerous" to startups is that with their substantial networks of users, they can institute changes such that your user base can be sucked away virtually overnight by little more than a press release and a beta version.

Regardless of the opposing company or the type of power it may wield, the situation is very similar and can generally be described as follow:. Your startup is "David" and their company is "Goliath". In some circles, this is referred to as a death sentence. When your startup stands in the way of a powerhouse who has resources that it can bring to bear on defeating you, there is often little you can do.

Something to keep in mind is that "often" does not mean "always". There are a few very important strategies that you can use to compete effectively against a larger competitor. Many of these strategies will remain valid whether you are each going after the same market, or if you have decided that you are going to go on the offensive because the product "Y" from company "X" is so abysmal, that its users can't possibly withstand the pain much longer.

Strategy 1: Fly under their radar

This is a short term strategy that you can use to survive for a while. It's not perfect, and it does have some flaws. For example, it won't work very effectively if your opponent sees this market as critical to their success. Trying to tackle Google and build a better search engine is almost doomed to failure. Just ask the Bing team at Microsoft. They have billions of dollars at their disposal and to date, have still come up short.

It's also not going to work well if you are both new to the market with your product, because you can bet your favorite de-motivational poster that they're already on the lookout for competitors. When they find you, they will actively seek to copy the features of your product and beat you in the market.

If your competitor knows who you are and you know that they know, look for another strategy because it's too late.

However, if you're the upstart in the industry and it's a relatively mature industry with several enterprise level vendors selling into it, this will probably work quite well. In Enterprise sales, mature products are handed off to teams of people to sell who consistently don't get any real work done. This is not to say that they don't make sales, because they do. Their problem is that they become so entrenched in the status quo, that eventually they lose all respect for their existing competitors and don't realize that an upstart like you might have a shot in the market, especially if you address their shortcomings.

It's not until you start taking multiple sales per quarter from the same regional sales reps that someone is going to notice anything is wrong. It could be months before

this happens, or even years, depending on the regional sales rep churn rate at your competitor. The time that this can buy you is crucial. An enterprise company will put out a new release, on average, once or twice per year. Occasionally they will do a quarterly release, but as a startup, you can beat them at the release game. You are more readily able to churn out a new release for each customer, as new demands come to the table and it takes more code to land each customer.

Strategy 2: Be where they aren't

If you've never heard of the technology marketing pyramid, don't worry about it for two reasons. First, I'm about to explain it to you, and second, I'm remembering it from a conversation I had with the VP of Sales from a company I worked for a long time ago. There's probably an official name for this, and if you know it, please drop me a line to fill me in. Otherwise, bear with me for a few minutes.

Any market for a high technology product can be divided into a few different segments within a pyramid. At the very top of the pyramid, you have the people who want or need the bleeding edge stuff, for whatever reason. The definition of what is considered to be bleeding edge is going to change based on the type of product. However it may make it easier to understand if we use a more concrete example.

Let's look at the database software market for a few minutes. When it comes to bleeding edge performance, whose name leaps to mind first: Microsoft or Oracle? Most people will think of Oracle first. Remember that we are discussing impressions of performance, not actual performance, nor are we discussing usability or pricing.

Oracle created their database software with the intent to run it on as many different platforms as they possibly could and wanted to do so with rock solid performance. Most people would agree that Oracle licenses are way overpriced and the way they sell their software is little better than the tactics employed by a used car salesman.

Customer: "How much for 4 Oracle licenses?"

Oracle: "Well, there are a lot of things that factor into it, including number of processors, types of processors, modules, software maintenance, how close I am to my quarterly bonus, etc. What's your budget for this project?"

Microsoft looked at how Oracle had positioned itself in the market as the elite database engine with the best performance you could possibly buy and on as many platforms as you could ever imagine. So what did Microsoft do? They shipped Microsoft Access. Many would argue it's not a true database, but then again, neither is Excel and people use it extensively as if it were. Hell, Microsoft even shipped an ODBC driver interface to get at the data inside of Excel.

But the point is that Microsoft chose to go where Oracle wasn't. Oracle was well known at the high end, but they were also known for being expensive. You get what you pay for, right?

Microsoft decided that if there was a small segment of the market at the top of the pyramid that were willing to pay Oracle for bleeding edge performance, then Oracle could have them. And Microsoft was going to take everything else. And so Microsoft did.

Microsoft Access went on to become one of the most prevalent desktop databases, a market position that it still dominates. Indirectly, they leveraged that success to assist with the establishment of SQL Server in the small to mid-level enterprise. But they still have issues pushing their database into markets where extreme performance or scalability is required. It's not that SQL Server doesn't work in these environments. There's simply the conception that it doesn't work well and is going to be a hassle to implement or somewhat unstable.

Over time, this strategy can also fall apart. As you can see in today's market for databases, there's a lot of ambiguity. Oracle has nowhere to go but down market, so they've offered free versions of Oracle to help attract developers and get into deals

they otherwise wouldn't. MySQL started at the bottom of the pyramid and is working its way up, pushing into Microsoft along the way.

Microsoft rests in the middle. It is attempting to push higher by releasing high end versions of SQL Server to compete with Oracle while at the same time trying to fend off MySQL with free offerings on the low end.

It may be a long way of saying it, but the point is that regardless of facts, you can choose how to position your product in the market and you must differentiate yourself from your competitor, even if the differences are minimal.

This means that if they are pitching performance, you pitch reliability, or scalability, or something else. You can't also pitch performance after they have started that marketing effort. It just doesn't work and customers will be skeptical of your claims. Market positioning has little to do with facts and more to do with perception. Use that to establish where you want to be, rather than to emphasize where you are.

Strategy 3: Exploit their weaknesses

Are you going to compete with a desktop application? Build a web app. Are you competing with a web app? Build a desktop app. Do you hear their customers complaining about something relatively major? Pick that as one of your marketing points and start taking away their customers.

The idea here is to find chinks in their armor and begin exploiting them for your own gains. Large companies often don't realize that things are going terribly wrong until it's too late. Then they simply buy out their competitor to regain the lead. Sounds great, doesn't it? See? I knew you were in this for the money.

Strategy 4: Find the users who are pissed off

This strategy plays with fire a little bit. It's usually pretty easy to find where people hang out who hate the product of your competitor. Just add "sucks" to the end of their company or product name and do a search. You're almost certain to find a forum that

tells you exactly why they suck. Or a URL redirection back to the company website because they were smart enough to buy it first.

Finding these communities does a few things for you. First of all, it serves as valuable market research. These people are more than happy to share why that product or company sucks and how things "should" be done. Take the good, and leave the bad because let's face it. Customers don't always know what they're talking about.

Second, it gives you a place to market your ideas. If these people really hate the other product or company so much, chances are good they would be willing to defect and use yours instead, especially if it addresses their issues.

And finally, people who a pissed off tend to vent online... A lot. And they're more than happy to tell all their buddies about this great new product from a competitor of their old vendor that is ten times better and here are all the reasons why. Basically you have a built in evangelist network you can tap into.

But there is also fire to contend with. As I already mentioned, these vocal customers have a tendency to believe they know how a product should be built and what it should do. Sometimes, they are in the distinct minority. Worse still, sometimes they don't even know it. So before you go pitching your product to these people, make sure you can meet their demands and that they are at least reasonably justified in their requirements.

Just be cautions. The fact of the matter is that they are going to be a fickle bunch. They'll give you a lot of slack in the beginning. As your product matures, they will expect your product to match all of their needs. When it doesn't, you will have to leave them behind and accept that they may very well be on another forum denouncing your product. Chances are it won't be nearly as loud or nearly as vocal because they tend to feel like large companies deserve their full wrath and they should save that wrath for them... As if they can't simply create more.

Strategy 5: Integrate with third party software or develop a plug-in API

Enterprise companies don't tend to play well with others, especially when it doesn't suit them to do so. However, building integration points with other products can be the bread and butter of your business strategy. When an Enterprise company starts losing deals because they don't have a good story to tell around a type of product such as help desk software, they'll do the only thing they know how to do: go out and buy a company that sells help desk software.

They won't build their own. When is the last time you saw an Enterprise company release a new product? I know there are a few examples, but they are the exception rather than the rule. The reason for this is that is time consuming and expensive to build a new product from scratch. Even worse, from the standpoint of the Enterprise Company, building a new product is exceptionally risky.

What if they build something that customers don't actually want because they misread the market? What if they find it too difficult to establish solid market penetration because of the entrenched players? What if they launch a minimally functional product and they tarnish their reputation as producing bad software? Anything involving a tarnished reputation takes a long time to overcome. More often that tarnished reputation hangs like an albatross around their necks which is difficult, if not impossible to shed.

So instead, they trade money for the time it would take to get a foothold into the market. In doing so, they get the code, the market penetration, complete rights, all the future benefits of the product, and a new source of potential customers to peddle their existing product line to. You obviously don't have that option.

Instead, by providing integration points between your software and the products of other, preferably larger vendors, you can gain a steady stream of customers from those other products who are looking for integration points with products that they already use today. As this third party product grows, the potential for your revenue to increase

grows as well. Building integration points into multiple products can compound this growth.

The disadvantage to this is that there must be a valid reason that users of this third party product would use it in conjunction with yours. Not all third party products will be a good fit. In addition, you have little to no control over those products. If they decide to go in a different direction, change their API, or eliminate it altogether, there is little you can do about it.

Finally, you must provide solid value to the customers via the integration points you provide because the customers must now pay for multiple products. Depending on the price point, this could be a difficult pill for customers to swallow.

Final Thoughts

Finding companies who have successfully employed one or more of these strategies isn't terribly difficult. Certain open source products such as Linux and MySQL have certainly carved out rather large portions of the market. On the commercial side, you don't have to look any further than companies like Mint.com, Skype (which took on traditional phone providers), and Netflix. For smaller, self-funded companies you can look to Red Gate, 37signals, Source Gear, and Atlassian.

To people like us, these companies are financially doing quite well today. But they all had to start at the bottom and cut their way through the competition before they got to where they are today. That takes time, effort, and a lot of hard work.

Conclusion

There's a lot more where this came from – don't miss MicroConf 2011 to hear more from these (and other) successful founders: *www.MicroConf.com*

If you enjoyed this eBook, please let us know on Twitter:

@microconf Loved your free 'Startup Wisdom' eBook: http://bit.ly/MicroConf2