Table of Contents

- Inequality and Risk
- What I Did this Summer
- Ideas for Startups
- The Venture Capital Squeeze
- How to Fund a Startup
- Web 2.0
- Good and Bad Procrastination
- How to Do What You Love
- Why YC
- 6,631,372
- Are Software Patents Evil?
- See Randomness
- The Hardest Lessons for Startups to Learn
- How to Be Silicon Valley
- Why Startups Condense in America
- The Power of the Marginal
- The Island Test
- Copy What You Like
- How to Present to Investors
- A Student's Guide to Startups
- The 18 Mistakes That Kill Startups
- How Art Can Be Good
- Learning from Founders
- Is It Worth Being Wise?
- Why to Not Not Start a Startup
- Microsoft is Dead
- Two Kinds of Judgement
- The Hacker's Guide to Investors
- An Alternative Theory of Unions
- The Equity Equation
- Stuff
- Holding a Program in One's Head
- How Not to Die
- News from the Front
- How to Do Philosophy
- The Future of Web Startups
- Why to Move to a Startup Hub
- Six Principles for Making New Things
- Trolls
- A New Venture Animal
- You Weren't Meant to Have a Boss
- How to Disagree
- Some Heroes
- Why There Aren't More Googles
- Be Good
- Lies We Tell Kids
- Disconnecting Distraction

- Cities and Ambition
- The Pooled-Risk Company Management Company
- A Fundraising Survival Guide

Inequality and Risk

August 2005

(This essay is derived from a talk at Defcon 2005.)

Suppose you wanted to get rid of economic inequality. There are two ways to do it: give money to the poor, or take it away from the rich. But they amount to the same thing, because if you want to give money to the poor, you have to get it from somewhere. You can't get it from the poor, or they just end up where they started. You have to get it from the rich.

There is of course a way to make the poor richer without simply shifting money from the rich. You could help the poor become more productive — for example, by improving access to education. Instead of taking money from engineers and giving it to checkout clerks, you could enable people who would have become checkout clerks to become engineers.

This is an excellent strategy for making the poor richer. But the evidence of the last 200 years shows that it doesn't reduce economic inequality, because it makes the rich richer too. If there are more engineers, then there are more opportunities to hire them and to sell them things. Henry Ford couldn't have made a fortune building cars in a society in which most people were still subsistence farmers; he would have had neither workers nor customers.

If you want to reduce economic inequality instead of just improving the overall standard of living, it's not enough just to raise up the poor. What if one of your newly minted engineers gets ambitious and goes on to become another Bill Gates? Economic inequality will be as bad as ever. If you actually want to compress the gap between rich and poor, you have to push down on the top as well as pushing up on the bottom.

How do you push down on the top? You could try to decrease the productivity of the people who make the most money: make the best surgeons operate with their left hands, force popular actors to overeat, and so on. But this approach is hard to implement. The only practical solution is to let people do the best work they can, and then (either by taxation or by limiting what they can charge) to confiscate whatever you deem to be surplus.

So let's be clear what reducing economic inequality means. It is identical with

taking money from the rich.

When you transform a mathematical expression into another form, you often notice new things. So it is in this case. Taking money from the rich turns out to have consequences one might not foresee when one phrases the same idea in terms of "reducing inequality."

The problem is, risk and reward have to be proportionate. A bet with only a 10% chance of winning has to pay more than one with a 50% chance of winning, or no one will take it. So if you lop off the top of the possible rewards, you thereby decrease people's willingness to take risks.

Transposing into our original expression, we get: decreasing economic inequality means decreasing the risk people are willing to take.

There are whole classes of risks that are no longer worth taking if the maximum return is decreased. One reason high tax rates are disastrous is that this class of risks includes starting new companies.

Investors

Startups are intrinsically risky. A startup is like a small boat in the open sea. One big wave and you're sunk. A competing product, a downturn in the economy, a delay in getting funding or regulatory approval, a patent suit, changing technical standards, the departure of a key employee, the loss of a big account — any one of these can destroy you overnight. It seems only about 1 in 10 startups succeeds. [1]

Our startup paid its first round of outside investors 36x. Which meant, with current US tax rates, that it made sense to invest in us if we had better than a 1 in 24 chance of succeeding. That sounds about right. That's probably roughly how we looked when we were a couple of nerds with no business experience operating out of an apartment.

If that kind of risk doesn't pay, venture investing, as we know it, doesn't happen.

That might be ok if there were other sources of capital for new companies. Why not just have the government, or some large almost-government organization like Fannie Mae, do the venture investing instead of private funds?

I'll tell you why that wouldn't work. Because then you're asking government or almost-government employees to do the one thing they are least able to do: take risks.

As anyone who has worked for the government knows, the important thing is not to make the right choices, but to make choices that can be justified later if they fail. If there is a safe option, that's the one a bureaucrat will choose. But that is exactly the wrong way to do venture investing. The nature of the business means

that you want to make terribly risky choices, if the upside looks good enough.

VCs are currently paid in a way that makes them focus on the upside: they get a percentage of the fund's gains. And that helps overcome their understandable fear of investing in a company run by nerds who look like (and perhaps are) college students.

If VCs weren't allowed to get rich, they'd behave like bureaucrats. Without hope of gain, they'd have only fear of loss. And so they'd make the wrong choices. They'd turn down the nerds in favor of the smooth-talking MBA in a suit, because that investment would be easier to justify later if it failed.

Founders

But even if you could somehow redesign venture funding to work without allowing VCs to become rich, there's another kind of investor you simply cannot replace: the startups' founders and early employees.

What they invest is their time and ideas. But these are equivalent to money; the proof is that investors are willing (if forced) to treat them as interchangeable, granting the same status to "sweat equity" and the equity they've purchased with cash.

The fact that you're investing time doesn't change the relationship between risk and reward. If you're going to invest your time in something with a small chance of succeeding, you'll only do it if there is a proportionately large payoff. [2] If large payoffs aren't allowed, you may as well play it safe.

Like many startup founders, I did it to get rich. But not because I wanted to buy expensive things. What I wanted was security. I wanted to make enough money that I didn't have to worry about money. If I'd been forbidden to make enough from a startup to do this, I would have sought security by some other means: for example, by going to work for a big, stable organization from which it would be hard to get fired. Instead of busting my ass in a startup, I would have tried to get a nice, low-stress job at a big research lab, or tenure at a university.

That's what everyone does in societies where risk isn't rewarded. If you can't ensure your own security, the next best thing is to make a nest for yourself in some large organization where your status depends mostly on seniority. [3]

Even if we could somehow replace investors, I don't see how we could replace founders. Investors mainly contribute money, which in principle is the same no matter what the source. But the founders contribute ideas. You can't replace those.

Let's rehearse the chain of argument so far. I'm heading for a conclusion to which many readers will have to be dragged kicking and screaming, so I've tried to make each link unbreakable. Decreasing economic inequality means taking money from the rich. Since risk and reward are equivalent, decreasing potential rewards

automatically decreases people's appetite for risk. Startups are intrinsically risky. Without the prospect of rewards proportionate to the risk, founders will not invest their time in a startup. Founders are irreplaceable. So eliminating economic inequality means eliminating startups.

Economic inequality is not just a consequence of startups. It's the engine that drives them, in the same way a fall of water drives a water mill. People start startups in the hope of becoming much richer than they were before. And if your society tries to prevent anyone from being much richer than anyone else, it will also prevent one person from being much richer at t2 than t1.

Growth

This argument applies proportionately. It's not just that if you eliminate economic inequality, you get no startups. To the extent you reduce economic inequality, you decrease the number of startups. [4] Increase taxes, and willingness to take risks decreases in proportion.

And that seems bad for everyone. New technology and new jobs both come disproportionately from new companies. Indeed, if you don't have startups, pretty soon you won't have established companies either, just as, if you stop having kids, pretty soon you won't have any adults.

It sounds benevolent to say we ought to reduce economic inequality. When you phrase it that way, who can argue with you? *Inequality* has to be bad, right? It sounds a good deal less benevolent to say we ought to reduce the rate at which new companies are founded. And yet the one implies the other.

Indeed, it may be that reducing investors' appetite for risk doesn't merely kill off larval startups, but kills off the most promising ones especially. Startups yield faster growth at greater risk than established companies. Does this trend also hold among startups? That is, are the riskiest startups the ones that generate most growth if they succeed? I suspect the answer is yes. And that's a chilling thought, because it means that if you cut investors' appetite for risk, the most beneficial startups are the first to go.

Not all rich people got that way from startups, of course. What if we let people get rich by starting startups, but taxed away all other surplus wealth? Wouldn't that at least decrease inequality?

Less than you might think. If you made it so that people could only get rich by starting startups, people who wanted to get rich would all start startups. And that might be a great thing. But I don't think it would have much effect on the distribution of wealth. People who want to get rich will do whatever they have to. If startups are the only way to do it, you'll just get far more people starting startups. (If you write the laws very carefully, that is. More likely, you'll just get a lot of people doing things that can be made to look on paper like startups.)

If we're determined to eliminate economic inequality, there is still one way out: we could say that we're willing to go ahead and do without startups. What would happen if we did?

At a minimum, we'd have to accept lower rates of technological growth. If you believe that large, established companies could somehow be made to develop new technology as fast as startups, the ball is in your court to explain how. (If you can come up with a remotely plausible story, you can make a fortune writing business books and consulting for large companies.) [5]

Ok, so we get slower growth. Is that so bad? Well, one reason it's bad in practice is that other countries might not agree to slow down with us. If you're content to develop new technologies at a slower rate than the rest of the world, what happens is that you don't invent anything at all. Anything you might discover has already been invented elsewhere. And the only thing you can offer in return is raw materials and cheap labor. Once you sink that low, other countries can do whatever they like with you: install puppet governments, siphon off your best workers, use your women as prostitutes, dump their toxic waste on your territory — all the things we do to poor countries now. The only defense is to isolate yourself, as communist countries did in the twentieth century. But the problem then is, you have to become a police state to enforce it.

Wealth and Power

I realize startups are not the main target of those who want to eliminate economic inequality. What they really dislike is the sort of wealth that becomes self-perpetuating through an alliance with power. For example, construction firms that fund politicians' campaigns in return for government contracts, or rich parents who get their children into good colleges by sending them to expensive schools designed for that purpose. But if you try to attack this type of wealth through *economic* policy, it's hard to hit without destroying startups as collateral damage.

The problem here is not wealth, but corruption. So why not go after corruption?

We don't need to prevent people from being rich if we can prevent wealth from translating into power. And there has been progress on that front. Before he died of drink in 1925, Commodore Vanderbilt's wastrel grandson Reggie ran down pedestrians on five separate occasions, killing two of them. By 1969, when Ted Kennedy drove off the bridge at Chappaquiddick, the limit seemed to be down to one. Today it may well be zero. But what's changed is not variation in wealth. What's changed is the ability to translate wealth into power.

How do you break the connection between wealth and power? Demand transparency. Watch closely how power is exercised, and demand an account of how decisions are made. Why aren't all police interrogations videotaped? Why did 36% of Princeton's class of 2007 come from prep schools, when only 1.7% of American kids attend them? Why did the US really invade Iraq? Why don't government officials disclose more about their finances, and why only during their

term of office?

A friend of mine who knows a lot about computer security says the single most important step is to log everything. Back when he was a kid trying to break into computers, what worried him most was the idea of leaving a trail. He was more inconvenienced by the need to avoid that than by any obstacle deliberately put in his path.

Like all illicit connections, the connection between wealth and power flourishes in secret. Expose all transactions, and you will greatly reduce it. Log everything. That's a strategy that already seems to be working, and it doesn't have the side effect of making your whole country poor.

I don't think many people realize there is a connection between economic inequality and risk. I didn't fully grasp it till recently. I'd known for years of course that if one didn't score in a startup, the other alternative was to get a cozy, tenured research job. But I didn't understand the equation governing my behavior. Likewise, it's obvious empirically that a country that doesn't let people get rich is headed for disaster, whether it's Diocletian's Rome or Harold Wilson's Britain. But I did not till recently understand the role risk played.

If you try to attack wealth, you end up nailing risk as well, and with it growth. If we want a fairer world, I think we're better off attacking one step downstream, where wealth turns into power.

Notes

[1] Success here is defined from the initial investors' point of view: either an IPO, or an acquisition for more than the valuation at the last round of funding. The conventional 1 in 10 success rate is suspiciously neat, but conversations with VCs suggest it's roughly correct for startups overall. Top VC firms expect to do better.

[2] I'm not claiming founders sit down and calculate the expected after-tax return from a startup. They're motivated by examples of other people who did it. And those examples do reflect after-tax returns.

[3] Conjecture: The variation in wealth in a (non-corrupt) country or organization will be inversely proportional to the prevalence of systems of seniority. So if you suppress variation in wealth, seniority will become correspondingly more important. So far, I know of no counterexamples, though in very corrupt countries you may get both simultaneously. (Thanks to Daniel Sobral for pointing this out.)

[4] In a country with a truly feudal economy, you might be able to redistribute wealth successfully, because there are no startups to kill.

[5] The speed at which startups develop new techology is the other reason they

pay so well. As I explained in "How to Make Wealth", what you do in a startup is compress a lifetime's worth of work into a few years. It seems as dumb to discourage that as to discourage risk-taking.

Thanks to Chris Anderson, Trevor Blackwell, Dan Giffin, Jessica Livingston, and Evan Williams for reading drafts of this essay, and to Langley Steinert, Sangam Pant, and Mike Moritz for information about venture investing.

Romanian Translation

Dutch Translation

Traditional Chinese Translation

Japanese Translation

Hebrew Translation

If you liked this, you may also like ***Hackers & Painters***.

What I Did this Summer

October 2005

The first Summer Founders Program has just finished. We were surprised how well it went. Overall only about 10% of startups succeed, but if I had to guess now, I'd predict three or four of the eight startups we funded will make it.

Of the startups that needed further funding, I believe all have either closed a round or are likely to soon. Two have already turned down (lowball) acquisition offers.

We would have been happy if just one of the eight seemed promising by the end of the summer. What's going on? Did some kind of anomaly make this summer's applicants especially good? We worry about that, but we can't think of one. We'll find out this winter.

The whole summer was full of surprises. The best was that the hypothesis we were testing seems to be correct. Young hackers can start viable companies. This is good news for two reasons: (a) it's an encouraging thought, and (b) it means that Y Combinator, which is predicated on the idea, is not hosed.

Age

More precisely, the hypothesis was that success in a startup depends mainly on

how smart and energetic you are, and much less on how old you are or how much business experience you have. The results so far bear this out. The 2005 summer founders ranged in age from 18 to 28 (average 23), and there is no correlation between their ages and how well they're doing.

This should not really be surprising. Bill Gates and Michael Dell were both 19 when they started the companies that made them famous. Young founders are not a new phenomenon: the trend began as soon as computers got cheap enough for college kids to afford them.

Another of our hypotheses was that you can start a startup on less money than most people think. Other investors were surprised to hear the most we gave any group was $20,000. But we knew it was possible to start on that little because we started Viaweb on $10,000.

And so it proved this summer. Three months' funding is enough to get into second gear. We had a demo day for potential investors ten weeks in, and seven of the eight groups had a prototype ready by that time. One, Reddit, had already launched, and were able to give a demo of their live site.

A researcher who studied the SFP startups said the one thing they had in common was that they all worked ridiculously hard. People this age are commonly seen as lazy. I think in some cases it's not so much that they lack the appetite for work, but that the work they're offered is unappetizing.

The experience of the SFP suggests that if you let motivated people do real work, they work hard, whatever their age. As one of the founders said "I'd read that starting a startup consumed your life, but I had no idea what that meant until I did it."

I'd feel guilty if I were a boss making people work this hard. But we're not these people's bosses. They're working on their own projects. And what makes them work is not us but their competitors. Like good athletes, they don't work hard because the coach yells at them, but because they want to win.

We have less power than bosses, and yet the founders work harder than employees. It seems like a win for everyone. The only catch is that we get on average only about 5-7% of the upside, while an employer gets nearly all of it. (We're counting on it being 5-7% of a much larger number.)

As well as working hard, the groups all turned out to be extraordinarily responsible. I can't think of a time when one failed to do something they'd promised to, even by being late for an appointment. This is another lesson the world has yet to learn. One of the founders discovered that the hardest part of arranging a meeting with executives at a big cell phone carrier was getting a rental company to rent him a car, because he was too young.

I think the problem here is much the same as with the apparent laziness of people

this age. They seem lazy because the work they're given is pointless, and they act irresponsible because they're not given any power. Some of them, anyway. We only have a sample size of about twenty, but it seems so far that if you let people in their early twenties be their own bosses, they rise to the occasion.

Morale

The summer founders were as a rule very idealistic. They also wanted very much to get rich. These qualities might seem incompatible, but they're not. These guys want to get rich, but they want to do it by changing the world. They wouldn't (well, seven of the eight groups wouldn't) be interested in making money by speculating in stocks. They want to make something people use.

I think this makes them more effective as founders. As hard as people will work for money, they'll work harder for a cause. And since success in a startup depends so much on motivation, the paradoxical result is that the people likely to make the most money are those who aren't in it just for the money.

The founders of Kiko, for example, are working on an Ajax calendar. They want to get rich, but they pay more attention to design than they would if that were their only motivation. You can tell just by looking at it.

I never considered it till this summer, but this might be another reason startups run by hackers tend to do better than those run by MBAs. Perhaps it's not just that hackers understand technology better, but that they're driven by more powerful motivations. Microsoft, as I've said before, is a dangerously misleading example. Their mean corporate culture only works for monopolies. Google is a better model.

Considering that the summer founders are the sharks in this ocean, we were surprised how frightened most of them were of competitors. But now that I think of it, we were just as frightened when we started Viaweb. For the first year, our initial reaction to news of a competitor was always: we're doomed. Just as a hypochondriac magnifies his symptoms till he's convinced he has some terrible disease, when you're not used to competitors you magnify them into monsters.

Here's a handy rule for startups: competitors are rarely as dangerous as they seem. Most will self-destruct before you can destroy them. And it certainly doesn't matter how many of them there are, any more than it matters to the winner of a marathon how many runners are behind him.

"It's a crowded market," I remember one founder saying worriedly.

"Are you the current leader?" I asked.

"Yes."

"Is anyone able to develop software faster than you?"

"Probably not."

"Well, if you're ahead now, and you're the fastest, then you'll stay ahead. What difference does it make how many others there are?"

Another group was worried when they realized they had to rewrite their software from scratch. I told them it would be a bad sign if they didn't. The main function of your initial version is to be rewritten.

That's why we advise groups to ignore issues like scalability, internationalization, and heavy-duty security at first. [1] I can imagine an advocate of "best practices" saying these ought to be considered from the start. And he'd be right, except that they interfere with the primary function of software in a startup: to be a vehicle for experimenting with its own design. Having to retrofit internationalization or scalability is a pain, certainly. The only bigger pain is not needing to, because your initial version was too big and rigid to evolve into something users wanted.

I suspect this is another reason startups beat big companies. Startups can be irresponsible and release version 1s that are light enough to evolve. In big companies, all the pressure is in the direction of over-engineering.

What Got Learned

One thing we were curious about this summer was where these groups would need help. That turned out to vary a lot. Some we helped with technical advice-- for example, about how to set up an application to run on multiple servers. Most we helped with strategy questions, like what to patent, and what to charge for and what to give away. Nearly all wanted advice about dealing with future investors: how much money should they take and what kind of terms should they expect?

However, all the groups quickly learned how to deal with stuff like patents and investors. These problems aren't intrinsically difficult, just unfamiliar.

It was surprising-- slightly frightening even-- how fast they learned. The weekend before the demo day for investors, we had a practice session where all the groups gave their presentations. They were all terrible. We tried to explain how to make them better, but we didn't have much hope. So on demo day I told the assembled angels and VCs that these guys were hackers, not MBAs, and so while their software was good, we should not expect slick presentations from them.

The groups then proceeded to give fabulously slick presentations. Gone were the mumbling recitations of lists of features. It was as if they'd spent the past week at acting school. I still don't know how they did it.

Perhaps watching each others' presentations helped them see what they'd been doing wrong. Just as happens in college, the summer founders learned a lot from one another-- maybe more than they learned from us. A lot of the problems they face are the same, from dealing with investors to hacking Javascript.

I don't want to give the impression there were no problems this summer. A lot went wrong, as usually happens with startups. One group got an "exploding term-sheet" from some VCs. Pretty much all the groups who had dealings with big companies found that big companies do everything infinitely slowly. (This is to be expected. If big companies weren't incapable, there would be no room for startups to exist.) And of course there were the usual nightmares associated with servers.

In short, the disasters this summer were just the usual childhood diseases. Some of this summer's eight startups will probably die eventually; it would be extraordinary if all eight succeeded. But what kills them will not be dramatic, external threats, but a mundane, internal one: not getting enough done.

So far, though, the news is all good. In fact, we were surprised how much fun the summer was for us. The main reason was how much we liked the founders. They're so earnest and hard-working. They seem to like us too. And this illustrates another advantage of investing over hiring: our relationship with them is way better than it would be between a boss and an employee. Y Combinator ends up being more like an older brother than a parent.

I was surprised how much time I spent making introductions. Fortunately I discovered that when a startup needed to talk to someone, I could usually get to the right person by at most one hop. I remember wondering, how did my friends get to be so eminent? and a second later realizing: shit, I'm forty.

Another surprise was that the three-month batch format, which we were forced into by the constraints of the summer, turned out to be an advantage. When we started Y Combinator, we planned to invest the way other venture firms do: as proposals came in, we'd evaluate them and decide yes or no. The SFP was just an experiment to get things started. But it worked so well that we plan to do all our investing this way, one cycle in the summer and one in winter. It's more efficient for us, and better for the startups too.

Several groups said our weekly dinners saved them from a common problem afflicting startups: working so hard that one has no social life. (I remember that part all too well.) This way, they were guaranteed a social event at least once a week.

Independence

I've heard Y Combinator described as an "incubator." Actually we're the opposite: incubators exert more control than ordinary VCs, and we make a point of exerting less. Among other things, incubators usually make you work in their office-- that's where the word "incubator" comes from. That seems the wrong model. If investors get too involved, they smother one of the most powerful forces in a startup: the feeling that it's your own company.

Incubators were conspicuous failures during the Bubble. There's still debate about

whether this was because of the Bubble, or because they're a bad idea. My vote is they're a bad idea. I think they fail because they select for the wrong people. When we were starting a startup, we would never have taken funding from an "incubator." We can find office space, thanks; just give us the money. And people with that attitude are the ones likely to succeed in startups.

Indeed, one quality all the founders shared this summer was a spirit of independence. I've been wondering about that. Are some people just a lot more independent than others, or would everyone be this way if they were allowed to?

As with most nature/nurture questions, the answer is probably: some of each. But my main conclusion from the summer is that there's more environment in the mix than most people realize. I could see that from how the founders' attitudes *changed* during the summer. Most were emerging from twenty or so years of being told what to do. They seemed a little surprised at having total freedom. But they grew into it really quickly; some of these guys now seem about four inches taller (metaphorically) than they did at the beginning of the summer.

When we asked the summer founders what surprised them most about starting a company, one said "the most shocking thing is that it worked."

It will take more experience to know for sure, but my guess is that a lot of hackers could do this-- that if you put people in a position of independence, they develop the qualities they need. Throw them off a cliff, and most will find on the way down that they have wings.

The reason this is news to anyone is that the same forces work in the other direction too. Most hackers are employees, and this molds you into someone to whom starting a startup seems impossible as surely as starting a startup molds you into someone who can handle it.

If I'm right, "hacker" will mean something different in twenty years than it does now. Increasingly it will mean the people who run the company. Y Combinator is just accelerating a process that would have happened anyway. Power is shifting from the people who deal with money to the people who create technology, and if our experience this summer is any guide, this will be a good thing.

Notes

[1] By heavy-duty security I mean efforts to protect against truly determined attackers.

The image shows us, the 2005 summer founders, and Smartleaf co-founders Mark Nitzberg and Olin Shivers at the 30-foot table Kate Courteau designed for us. Photo by Alex Lewin.

Thanks to Sarah Harlin, Steve Huffman, Jessica Livingston, Zak Stone, and Aaron Swartz for reading drafts of this.

Romanian Translation

Japanese Translation

Ideas for Startups

October 2005

(This essay is derived from a talk at the 2005 Startup School.)

How do you get good ideas for startups? That's probably the number one question people ask me.

I'd like to reply with another question: why do people think it's hard to come up with ideas for startups?

That might seem a stupid thing to ask. Why do they *think* it's hard? If people can't do it, then it *is* hard, at least for them. Right?

Well, maybe not. What people usually say is not that they can't think of ideas, but that they don't have any. That's not quite the same thing. It could be the reason they don't have any is that they haven't tried to generate them.

I think this is often the case. I think people believe that coming up with ideas for startups is very hard-- that it *must* be very hard-- and so they don't try do to it. They assume ideas are like miracles: they either pop into your head or they don't.

I also have a theory about why people think this. They overvalue ideas. They think creating a startup is just a matter of implementing some fabulous initial idea. And since a successful startup is worth millions of dollars, a good idea is therefore a million dollar idea.

If coming up with an idea for a startup equals coming up with a million dollar idea, then of course it's going to seem hard. Too hard to bother trying. Our instincts tell us something so valuable would not be just lying around for anyone to discover.

Actually, startup ideas are not million dollar ideas, and here's an experiment you can try to prove it: just try to sell one. Nothing evolves faster than markets. The fact that there's no market for startup ideas suggests there's no demand. Which means, in the narrow sense of the word, that startup ideas are worthless.

Questions

The fact is, most startups end up nothing like the initial idea. It would be closer to the truth to say the main value of your initial idea is that, in the process of discovering it's broken, you'll come up with your real idea.

The initial idea is just a starting point-- not a blueprint, but a question. It might help if they were expressed that way. Instead of saying that your idea is to make a collaborative, web-based spreadsheet, say: could one make a collaborative, web-based spreadsheet? A few grammatical tweaks, and a woefully incomplete idea becomes a promising question to explore.

There's a real difference, because an assertion provokes objections in a way a question doesn't. If you say: I'm going to build a web-based spreadsheet, then critics-- the most dangerous of which are in your own head-- will immediately reply that you'd be competing with Microsoft, that you couldn't give people the kind of UI they expect, that users wouldn't want to have their data on your servers, and so on.

A question doesn't seem so challenging. It becomes: let's try making a web-based spreadsheet and see how far we get. And everyone knows that if you tried this you'd be able to make *something* useful. Maybe what you'd end up with wouldn't even be a spreadsheet. Maybe it would be some kind of new spreadsheet-like collaboration tool that doesn't even have a name yet. You wouldn't have thought of something like that except by implementing your way toward it.

Treating a startup idea as a question changes what you're looking for. If an idea is a blueprint, it has to be right. But if it's a question, it can be wrong, so long as it's wrong in a way that leads to more ideas.

One valuable way for an idea to be wrong is to be only a partial solution. When someone's working on a problem that seems too big, I always ask: is there some way to bite off some subset of the problem, then gradually expand from there? That will generally work unless you get trapped on a local maximum, like 1980s-style AI, or C.

Upwind

So far, we've reduced the problem from thinking of a million dollar idea to thinking of a mistaken question. That doesn't seem so hard, does it?

To generate such questions you need two things: to be familiar with promising new technologies, and to have the right kind of friends. New technologies are the ingredients startup ideas are made of, and conversations with friends are the kitchen they're cooked in.

Universities have both, and that's why so many startups grow out of them. They're filled with new technologies, because they're trying to produce research, and only things that are new count as research. And they're full of exactly the right kind of people to have ideas with: the other students, who will be not only smart but

elastic-minded to a fault.

The opposite extreme would be a well-paying but boring job at a big company. Big companies are biased against new technologies, and the people you'd meet there would be wrong too.

In an essay I wrote for high school students, I said a good rule of thumb was to stay upwind-- to work on things that maximize your future options. The principle applies for adults too, though perhaps it has to be modified to: stay upwind for as long as you can, then cash in the potential energy you've accumulated when you need to pay for kids.

I don't think people consciously realize this, but one reason downwind jobs like churning out Java for a bank pay so well is precisely that they are downwind. The market price for that kind of work is higher because it gives you fewer options for the future. A job that lets you work on exciting new stuff will tend to pay less, because part of the compensation is in the form of the new skills you'll learn.

Grad school is the other end of the spectrum from a coding job at a big company: the pay's low but you spend most of your time working on new stuff. And of course, it's called "school," which makes that clear to everyone, though in fact all jobs are some percentage school.

The right environment for having startup ideas need not be a university per se. It just has to be a situation with a large percentage of school.

It's obvious why you want exposure to new technology, but why do you need other people? Can't you just think of new ideas yourself? The empirical answer is: no. Even Einstein needed people to bounce ideas off. Ideas get developed in the process of explaining them to the right kind of person. You need that resistance, just as a carver needs the resistance of the wood.

This is one reason Y Combinator has a rule against investing in startups with only one founder. Practically every successful company has at least two. And because startup founders work under great pressure, it's critical they be friends.

I didn't realize it till I was writing this, but that may help explain why there are so few female startup founders. I read on the Internet (so it must be true) that only 1.7% of VC-backed startups are founded by women. The percentage of female hackers is small, but not that small. So why the discrepancy?

When you realize that successful startups tend to have multiple founders who were already friends, a possible explanation emerges. People's best friends are likely to be of the same sex, and if one group is a minority in some population, *pairs* of them will be a minority squared. [1]

Doodling

What these groups of co-founders do together is more complicated than just sitting down and trying to think of ideas. I suspect the most productive setup is a kind of together-alone-together sandwich. Together you talk about some hard problem, probably getting nowhere. Then, the next morning, one of you has an idea in the shower about how to solve it. He runs eagerly to to tell the others, and together they work out the kinks.

What happens in that shower? It seems to me that ideas just pop into my head. But can we say more than that?

Taking a shower is like a form of meditation. You're alert, but there's nothing to distract you. It's in a situation like this, where your mind is free to roam, that it bumps into new ideas.

What happens when your mind wanders? It may be like doodling. Most people have characteristic ways of doodling. This habit is unconscious, but not random: I found my doodles changed after I started studying painting. I started to make the kind of gestures I'd make if I were drawing from life. They were atoms of drawing, but arranged randomly. [2]

Perhaps letting your mind wander is like doodling with ideas. You have certain mental gestures you've learned in your work, and when you're not paying attention, you keep making these same gestures, but somewhat randomly. In effect, you call the same functions on random arguments. That's what a metaphor is: a function applied to an argument of the wrong type.

Conveniently, as I was writing this, my mind wandered: would it be useful to have metaphors in a programming language? I don't know; I don't have time to think about this. But it's convenient because this is an example of what I mean by habits of mind. I spend a lot of time thinking about language design, and my habit of always asking "would x be useful in a programming language" just got invoked.

If new ideas arise like doodles, this would explain why you have to work at something for a while before you have any. It's not just that you can't judge ideas till you're an expert in a field. You won't even generate ideas, because you won't have any habits of mind to invoke.

Of course the habits of mind you invoke on some field don't have to be derived from working in that field. In fact, it's often better if they're not. You're not just looking for good ideas, but for good *new* ideas, and you have a better chance of generating those if you combine stuff from distant fields. As hackers, one of our habits of mind is to ask, could one open-source x? For example, what if you made an open-source operating system? A fine idea, but not very novel. Whereas if you ask, could you make an open-source play? you might be onto something.

Are some kinds of work better sources of habits of mind than others? I suspect harder fields may be better sources, because to attack hard problems you need powerful solvents. I find math is a good source of metaphors-- good enough that

it's worth studying just for that. Related fields are also good sources, especially when they're related in unexpected ways. Everyone knows computer science and electrical engineering are related, but precisely because everyone knows it, importing ideas from one to the other doesn't yield great profits. It's like importing something from Wisconsin to Michigan. Whereas (I claim) hacking and painting are also related, in the sense that hackers and painters are both makers, and this source of new ideas is practically virgin territory.

Problems

In theory you could stick together ideas at random and see what you came up with. What if you built a peer-to-peer dating site? Would it be useful to have an automatic book? Could you turn theorems into a commodity? When you assemble ideas at random like this, they may not be just stupid, but semantically ill-formed. What would it even mean to make theorems a commodity? You got me. I didn't think of that idea, just its name.

You might come up with something useful this way, but I never have. It's like knowing a fabulous sculpture is hidden inside a block of marble, and all you have to do is remove the marble that isn't part of it. It's an encouraging thought, because it reminds you there is an answer, but it's not much use in practice because the search space is too big.

I find that to have good ideas I need to be working on some problem. You can't start with randomness. You have to start with a problem, then let your mind wander just far enough for new ideas to form.

In a way, it's harder to see problems than their solutions. Most people prefer to remain in denial about problems. It's obvious why: problems are irritating. They're problems! Imagine if people in 1700 saw their lives the way we'd see them. It would have been unbearable. This denial is such a powerful force that, even when presented with possible solutions, people often prefer to believe they wouldn't work.

I saw this phenomenon when I worked on spam filters. In 2002, most people preferred to ignore spam, and most of those who didn't preferred to believe the heuristic filters then available were the best you could do.

I found spam intolerable, and I felt it had to be possible to recognize it statistically. And it turns out that was all you needed to solve the problem. The algorithm I used was ridiculously simple. Anyone who'd really tried to solve the problem would have found it. It was just that no one had really tried to solve the problem. [3]

Let me repeat that recipe: finding the problem intolerable and feeling it must be possible to solve it. Simple as it seems, that's the recipe for a lot of startup ideas.

Wealth

So far most of what I've said applies to ideas in general. What's special about startup ideas? Startup ideas are ideas for companies, and companies have to make money. And the way to make money is to make something people want.

Wealth is what people want. I don't mean that as some kind of philosophical statement; I mean it as a tautology.

So an idea for a startup is an idea for something people want. Wouldn't any good idea be something people want? Unfortunately not. I think new theorems are a fine thing to create, but there is no great demand for them. Whereas there appears to be great demand for celebrity gossip magazines. Wealth is defined democratically. Good ideas and valuable ideas are not quite the same thing; the difference is individual tastes.

But valuable ideas are very close to good ideas, especially in technology. I think they're so close that you can get away with working as if the goal were to discover good ideas, so long as, in the final stage, you stop and ask: will people actually pay for this? Only a few ideas are likely to make it that far and then get shot down; RPN calculators might be one example.

One way to make something people want is to look at stuff people use now that's broken. Dating sites are a prime example. They have millions of users, so they must be promising something people want. And yet they work horribly. Just ask anyone who uses them. It's as if they used the worse-is-better approach but stopped after the first stage and handed the thing over to marketers.

Of course, the most obvious breakage in the average computer user's life is Windows itself. But this is a special case: you can't defeat a monopoly by a frontal attack. Windows can and will be overthrown, but not by giving people a better desktop OS. The way to kill it is to redefine the problem as a superset of the current one. The problem is not, what operating system should people use on desktop computers? but how should people use applications? There are answers to that question that don't even involve desktop computers.

Everyone thinks Google is going to solve this problem, but it is a very subtle one, so subtle that a company as big as Google might well get it wrong. I think the odds are better than 50-50 that the Windows killer-- or more accurately, Windows transcender-- will come from some little startup.

Another classic way to make something people want is to take a luxury and make it into a commmodity. People must want something if they pay a lot for it. And it is a very rare product that can't be made dramatically cheaper if you try.

This was Henry Ford's plan. He made cars, which had been a luxury item, into a commodity. But the idea is much older than Henry Ford. Water mills transformed mechanical power from a luxury into a commodity, and they were used in the Roman empire. Arguably pastoralism transformed a luxury into a commodity.

When you make something cheaper you can sell more of them. But if you make something dramatically cheaper you often get qualitative changes, because people start to use it in different ways. For example, once computers get so cheap that most people can have one of their own, you can use them as communication devices.

Often to make something dramatically cheaper you have to redefine the problem. The Model T didn't have all the features previous cars did. It only came in black, for example. But it solved the problem people cared most about, which was getting from place to place.

One of the most useful mental habits I know I learned from Michael Rabin: that the best way to solve a problem is often to redefine it. A lot of people use this technique without being consciously aware of it, but Rabin was spectacularly explicit. You need a big prime number? Those are pretty expensive. How about if I give you a big number that only has a 10 to the minus 100 chance of not being prime? Would that do? Well, probably; I mean, that's probably smaller than the chance that I'm imagining all this anyway.

Redefining the problem is a particularly juicy heuristic when you have competitors, because it's so hard for rigid-minded people to follow. You can work in plain sight and they don't realize the danger. Don't worry about us. We're just working on search. Do one thing and do it well, that's our motto.

Making things cheaper is actually a subset of a more general technique: making things easier. For a long time it was most of making things easier, but now that the things we build are so complicated, there's another rapidly growing subset: making things easier to *use*.

This is an area where there's great room for improvement. What you want to be able to say about technology is: it just works. How often do you say that now?

Simplicity takes effort-- genius, even. The average programmer seems to produce UI designs that are almost willfully bad. I was trying to use the stove at my mother's house a couple weeks ago. It was a new one, and instead of physical knobs it had buttons and an LED display. I tried pressing some buttons I thought would cause it to get hot, and you know what it said? "Err." Not even "Error." "Err." You can't just say "Err" to the user of a *stove*. You should design the UI so that errors are impossible. And the boneheads who designed this stove even had an example of such a UI to work from: the old one. You turn one knob to set the temperature and another to set the timer. What was wrong with that? It just worked.

It seems that, for the average engineer, more options just means more rope to hang yourself. So if you want to start a startup, you can take almost any existing technology produced by a big company, and assume you could build something way easier to use.

Design for Exit

Success for a startup approximately equals getting bought. You need some kind of exit strategy, because you can't get the smartest people to work for you without giving them options likely to be worth something. Which means you either have to get bought or go public, and the number of startups that go public is very small.

If success probably means getting bought, should you make that a conscious goal? The old answer was no: you were supposed to pretend that you wanted to create a giant, public company, and act surprised when someone made you an offer. Really, you want to buy us? Well, I suppose we'd consider it, for the right price.

I think things are changing. If 98% of the time success means getting bought, why not be open about it? If 98% of the time you're doing product development on spec for some big company, why not think of that as your task? One advantage of this approach is that it gives you another source of ideas: look at big companies, think what they should be doing, and do it yourself. Even if they already know it, you'll probably be done faster.

Just be sure to make something multiple acquirers will want. Don't fix Windows, because the only potential acquirer is Microsoft, and when there's only one acquirer, they don't have to hurry. They can take their time and copy you instead of buying you. If you want to get market price, work on something where there's competition.

If an increasing number of startups are created to do product development on spec, it will be a natural counterweight to monopolies. Once some type of technology is captured by a monopoly, it will only evolve at big company rates instead of startup rates, whereas alternatives will evolve with especial speed. A free market interprets monopoly as damage and routes around it.

The Woz Route

The most productive way to generate startup ideas is also the most unlikely-sounding: by accident. If you look at how famous startups got started, a lot of them weren't initially supposed to be startups. Lotus began with a program Mitch Kapor wrote for a friend. Apple got started because Steve Wozniak wanted to build microcomputers, and his employer, Hewlett-Packard, wouldn't let him do it at work. Yahoo began as David Filo's personal collection of links.

This is not the only way to start startups. You can sit down and consciously come up with an idea for a company; we did. But measured in total market cap, the build-stuff-for-yourself model might be more fruitful. It certainly has to be the most fun way to come up with startup ideas. And since a startup ought to have multiple founders who were already friends before they decided to start a company, the rather surprising conclusion is that the best way to generate startup ideas is to do what hackers do for fun: cook up amusing hacks with your friends.

It seems like it violates some kind of conservation law, but there it is: the best way to get a "million dollar idea" is just to do what hackers enjoy doing anyway.

Notes

[1] This phenomenon may account for a number of discrepancies currently blamed on various forbidden isms. Never attribute to malice what can be explained by math.

[2] A lot of classic abstract expressionism is doodling of this type: artists trained to paint from life using the same gestures but without using them to represent anything. This explains why such paintings are (slightly) more interesting than random marks would be.

[3] Bill Yerazunis had solved the problem, but he got there by another path. He made a general-purpose file classifier so good that it also worked for spam.

One Specific Idea

Romanian Translation

Japanese Translation

Traditional Chinese Translation

Russian Translation

Arabic Translation

The Venture Capital Squeeze

November 2005

In the next few years, venture capital funds will find themselves squeezed from four directions. They're already stuck with a seller's market, because of the huge amounts they raised at the end of the Bubble and still haven't invested. This by itself is not the end of the world. In fact, it's just a more extreme version of the norm in the VC business: too much money chasing too few deals.

Unfortunately, those few deals now want less and less money, because it's getting so cheap to start a startup. The four causes: open source, which makes software free; Moore's law, which makes hardware geometrically closer to free; the Web, which makes promotion free if you're good; and better languages, which make development a lot cheaper.

When we started our startup in 1995, the first three were our biggest expenses. We had to pay $5000 for the Netscape Commerce Server, the only software that then supported secure http connections. We paid $3000 for a server with a 90 MHz processor and 32 meg of memory. And we paid a PR firm about $30,000 to promote our launch.

Now you could get all three for nothing. You can get the software for free; people throw away computers more powerful than our first server; and if you make something good you can generate ten times as much traffic by word of mouth online than our first PR firm got through the print media.

And of course another big change for the average startup is that programming languages have improved-- or rather, the median language has. At most startups ten years ago, software development meant ten programmers writing code in C++. Now the same work might be done by one or two using Python or Ruby.

During the Bubble, a lot of people predicted that startups would outsource their development to India. I think a better model for the future is David Heinemeier Hansson, who outsourced his development to a more powerful language instead. A lot of well-known applications are now, like BaseCamp, written by just one programmer. And one guy is more than 10x cheaper than ten, because (a) he won't waste any time in meetings, and (b) since he's probably a founder, he can pay himself nothing.

Because starting a startup is so cheap, venture capitalists now often want to give startups more money than the startups want to take. VCs like to invest several million at a time. But as one VC told me after a startup he funded would only take about half a million, "I don't know what we're going to do. Maybe we'll just have to give some of it back." Meaning give some of the fund back to the institutional investors who supplied it, because it wasn't going to be possible to invest it all.

Into this already bad situation comes the third problem: Sarbanes-Oxley. Sarbanes-Oxley is a law, passed after the Bubble, that drastically increases the regulatory burden on public companies. And in addition to the cost of compliance, which is at least two million dollars a year, the law introduces frightening legal exposure for corporate officers. An experienced CFO I know said flatly: "I would not want to be CFO of a public company now."

You might think that responsible corporate governance is an area where you can't go too far. But you can go too far in any law, and this remark convinced me that Sarbanes-Oxley must have. This CFO is both the smartest and the most upstanding money guy I know. If Sarbanes-Oxley deters people like him from being CFOs of public companies, that's proof enough that it's broken.

Largely because of Sarbanes-Oxley, few startups go public now. For all practical purposes, succeeding now equals getting bought. Which means VCs are now in the business of finding promising little 2-3 man startups and pumping them up into companies that cost $100 million to acquire. They didn't mean to be in this business; it's just what their business has evolved into.

Hence the fourth problem: the acquirers have begun to realize they can buy wholesale. Why should they wait for VCs to make the startups they want more expensive? Most of what the VCs add, acquirers don't want anyway. The acquirers already have brand recognition and HR departments. What they really want is the software and the developers, and that's what the startup is in the early phase: concentrated software and developers.

Google, typically, seems to have been the first to figure this out. "Bring us your startups early," said Google's speaker at the Startup School. They're quite explicit about it: they like to acquire startups at just the point where they would do a Series A round. (The Series A round is the first round of real VC funding; it usually happens in the first year.) It is a brilliant strategy, and one that other big technology companies will no doubt try to duplicate. Unless they want to have still more of their lunch eaten by Google.

Of course, Google has an advantage in buying startups: a lot of the people there are rich, or expect to be when their options vest. Ordinary employees find it very hard to recommend an acquisition; it's just too annoying to see a bunch of twenty year olds get rich when you're still working for salary. Even if it's the right thing for your company to do.

The Solution(s)

Bad as things look now, there is a way for VCs to save themselves. They need to do two things, one of which won't surprise them, and another that will seem an anathema.

Let's start with the obvious one: lobby to get Sarbanes-Oxley loosened. This law was created to prevent future Enrons, not to destroy the IPO market. Since the IPO market was practically dead when it passed, few saw what bad effects it would have. But now that technology has recovered from the last bust, we can see clearly what a bottleneck Sarbanes-Oxley has become.

Startups are fragile plants—seedlings, in fact. These seedlings are worth protecting, because they grow into the trees of the economy. Much of the economy's growth is their growth. I think most politicians realize that. But they don't realize just how fragile startups are, and how easily they can become collateral damage of laws meant to fix some other problem.

Still more dangerously, when you destroy startups, they make very little noise. If you step on the toes of the coal industry, you'll hear about it. But if you inadvertantly squash the startup industry, all that happens is that the founders of the next Google stay in grad school instead of starting a company.

My second suggestion will seem shocking to VCs: let founders cash out partially in the Series A round. At the moment, when VCs invest in a startup, all the stock they get is newly issued and all the money goes to the company. They could buy some stock directly from the founders as well.

Most VCs have an almost religious rule against doing this. They don't want founders to get a penny till the company is sold or goes public. VCs are obsessed with control, and they worry that they'll have less leverage over the founders if the founders have any money.

This is a dumb plan. In fact, letting the founders sell a little stock early would generally be better for the company, because it would cause the founders' attitudes toward risk to be aligned with the VCs'. As things currently work, their attitudes toward risk tend to be diametrically opposed: the founders, who have nothing, would prefer a 100% chance of $1 million to a 20% chance of $10 million, while the VCs can afford to be "rational" and prefer the latter.

Whatever they say, the reason founders are selling their companies early instead of doing Series A rounds is that they get paid up front. That first million is just worth so much more than the subsequent ones. If founders could sell a little stock early, they'd be happy to take VC money and bet the rest on a bigger outcome.

So why not let the founders have that first million, or at least half million? The VCs would get same number of shares for the money. So what if some of the money would go to the founders instead of the company?

Some VCs will say this is unthinkable—that they want all their money to be put to work growing the company. But the fact is, the huge size of current VC investments is dictated by the structure of VC funds, not the needs of startups. Often as not these large investments go to work destroying the company rather than growing it.

The angel investors who funded our startup let the founders sell some stock directly to them, and it was a good deal for everyone. The angels made a huge return on that investment, so they're happy. And for us founders it blunted the terrifying all-or-nothingness of a startup, which in its raw form is more a distraction than a motivator.

If VCs are frightened at the idea of letting founders partially cash out, let me tell them something still more frightening: you are now competing directly with Google.

Thanks to Trevor Blackwell, Sarah Harlin, Jessica Livingston, and Robert Morris for reading drafts of this.

Romanian Translation

Hebrew Translation

Japanese Translation

If you liked this, you may also like ***Hackers & Painters***.

How to Fund a Startup

November 2005

Venture funding works like gears. A typical startup goes through several rounds of funding, and at each round you want to take just enough money to reach the speed where you can shift into the next gear.

Few startups get it quite right. Many are underfunded. A few are overfunded, which is like trying to start driving in third gear.

I think it would help founders to understand funding better—not just the mechanics of it, but what investors are thinking. I was surprised recently when I realized that all the worst problems we faced in our startup were due not to competitors, but investors. Dealing with competitors was easy by comparison.

I don't mean to suggest that our investors were nothing but a drag on us. They were helpful in negotiating deals, for example. I mean more that conflicts with investors are particularly nasty. Competitors punch you in the jaw, but investors have you by the balls.

Apparently our situation was not unusual. And if trouble with investors is one of the biggest threats to a startup, managing them is one of the most important skills founders need to learn.

Let's start by talking about the five sources of startup funding. Then we'll trace the life of a hypothetical (very fortunate) startup as it shifts gears through successive rounds.

Friends and Family

A lot of startups get their first funding from friends and family. Excite did, for example: after the founders graduated from college, they borrowed $15,000 from their parents to start a company. With the help of some part-time jobs they made it last 18 months.

If your friends or family happen to be rich, the line blurs between them and angel investors. At Viaweb we got our first $10,000 of seed money from our friend Julian, but he was sufficiently rich that it's hard to say whether he should be classified as a friend or angel. He was also a lawyer, which was great, because it meant we didn't have to pay legal bills out of that initial small sum.

The advantage of raising money from friends and family is that they're easy to find. You already know them. There are three main disadvantages: you mix

together your business and personal life; they will probably not be as well connected as angels or venture firms; and they may not be accredited investors, which could complicate your life later.

The SEC defines an "accredited investor" as someone with over a million dollars in liquid assets or an income of over $200,000 a year. The regulatory burden is much lower if a company's shareholders are all accredited investors. Once you take money from the general public you're more restricted in what you can do. [1]

A startup's life will be more complicated, legally, if any of the investors aren't accredited. In an IPO, it might not merely add expense, but change the outcome. A lawyer I asked about it said:

> When the company goes public, the SEC will carefully study all prior issuances of stock by the company and demand that it take immediate action to cure any past violations of securities laws. Those remedial actions can delay, stall or even kill the IPO.

Of course the odds of any given startup doing an IPO are small. But not as small as they might seem. A lot of startups that end up going public didn't seem likely to at first. (Who could have guessed that the company Wozniak and Jobs started in their spare time selling plans for microcomputers would yield one of the biggest IPOs of the decade?) Much of the value of a startup consists of that tiny probability multiplied by the huge outcome.

It wasn't because they weren't accredited investors that I didn't ask my parents for seed money, though. When we were starting Viaweb, I didn't know about the concept of an accredited investor, and didn't stop to think about the value of investors' connections. The reason I didn't take money from my parents was that I didn't want them to lose it.

Consulting

Another way to fund a startup is to get a job. The best sort of job is a consulting project in which you can build whatever software you wanted to sell as a startup. Then you can gradually transform yourself from a consulting company into a product company, and have your clients pay your development expenses.

This is a good plan for someone with kids, because it takes most of the risk out of starting a startup. There never has to be a time when you have no revenues. Risk and reward are usually proportionate, however: you should expect a plan that cuts the risk of starting a startup also to cut the average return. In this case, you trade decreased financial risk for increased risk that your company won't succeed as a startup.

But isn't the consulting company itself a startup? No, not generally. A company has to be more than small and newly founded to be a startup. There are millions of small businesses in America, but only a few thousand are startups. To be a startup, a company has to be a product business, not a service business. By which I mean not that it has to make something physical, but that it has to have one thing it sells to many people, rather than doing custom work for individual clients. Custom

work doesn't scale. To be a startup you need to be the band that sells a million copies of a song, not the band that makes money by playing at individual weddings and bar mitzvahs.

The trouble with consulting is that clients have an awkward habit of calling you on the phone. Most startups operate close to the margin of failure, and the distraction of having to deal with clients could be enough to put you over the edge. Especially if you have competitors who get to work full time on just being a startup.

So you have to be very disciplined if you take the consulting route. You have to work actively to prevent your company growing into a "weed tree," dependent on this source of easy but low-margin money. [2]

Indeed, the biggest danger of consulting may be that it gives you an excuse for failure. In a startup, as in grad school, a lot of what ends up driving you are the expectations of your family and friends. Once you start a startup and tell everyone that's what you're doing, you're now on a path labelled "get rich or bust." You now have to get rich, or you've failed.

Fear of failure is an extraordinarily powerful force. Usually it prevents people from starting things, but once you publish some definite ambition, it switches directions and starts working in your favor. I think it's a pretty clever piece of jiujitsu to set this irresistible force against the slightly less immovable object of becoming rich. You won't have it driving you if your stated ambition is merely to start a consulting company that you will one day morph into a startup.

An advantage of consulting, as a way to develop a product, is that you know you're making something at least one customer wants. But if you have what it takes to start a startup you should have sufficient vision not to need this crutch.

Angel Investors

Angels are individual rich people. The word was first used for backers of Broadway plays, but now applies to individual investors generally. Angels who've made money in technology are preferable, for two reasons: they understand your situation, and they're a source of contacts and advice.

The contacts and advice can be more important than the money. When del.icio.us took money from investors, they took money from, among others, Tim O'Reilly. The amount he put in was small compared to the VCs who led the round, but Tim is a smart and influential guy and it's good to have him on your side.

You can do whatever you want with money from consulting or friends and family. With angels we're now talking about venture funding proper, so it's time to introduce the concept of *exit strategy*. Younger would-be founders are often surprised that investors expect them either to sell the company or go public. The reason is that investors need to get their capital back. They'll only consider companies that have an exit strategy—meaning companies that could get bought

or go public.

This is not as selfish as it sounds. There are few large, private technology companies. Those that don't fail all seem to get bought or go public. The reason is that employees are investors too—of their time—and they want just as much to be able to cash out. If your competitors offer employees stock options that might make them rich, while you make it clear you plan to stay private, your competitors will get the best people. So the principle of an "exit" is not just something forced on startups by investors, but part of what it means to be a startup.

Another concept we need to introduce now is valuation. When someone buys shares in a company, that implicitly establishes a value for it. If someone pays $20,000 for 10% of a company, the company is in theory worth $200,000. I say "in theory" because in early stage investing, valuations are voodoo. As a company gets more established, its valuation gets closer to an actual market value. But in a newly founded startup, the valuation number is just an artifact of the respective contributions of everyone involved.

Startups often "pay" investors who will help the company in some way by letting them invest at low valuations. If I had a startup and Steve Jobs wanted to invest in it, I'd give him the stock for $10, just to be able to brag that he was an investor. Unfortunately, it's impractical (if not illegal) to adjust the valuation of the company up and down for each investor. Startups' valuations are supposed to rise over time. So if you're going to sell cheap stock to eminent angels, do it early, when it's natural for the company to have a low valuation.

Some angel investors join together in syndicates. Any city where people start startups will have one or more of them. In Boston the biggest is the Common Angels. In the Bay Area it's the Band of Angels. You can find groups near you through the Angel Capital Association. [3] However, most angel investors don't belong to these groups. In fact, the more prominent the angel, the less likely they are to belong to a group.

Some angel groups charge you money to pitch your idea to them. Needless to say, you should never do this.

One of the dangers of taking investment from individual angels, rather than through an angel group or investment firm, is that they have less reputation to protect. A big-name VC firm will not screw you too outrageously, because other founders would avoid them if word got out. With individual angels you don't have this protection, as we found to our dismay in our own startup. In many startups' lives there comes a point when you're at the investors' mercy—when you're out of money and the only place to get more is your existing investors. When we got into such a scrape, our investors took advantage of it in a way that a name-brand VC probably wouldn't have.

Angels have a corresponding advantage, however: they're also not bound by all the rules that VC firms are. And so they can, for example, allow founders to cash

out partially in a funding round, by selling some of their stock directly to the investors. I think this will become more common; the average founder is eager to do it, and selling, say, half a million dollars worth of stock will not, as VCs fear, cause most founders to be any less committed to the business.

The same angels who tried to screw us also let us do this, and so on balance I'm grateful rather than angry. (As in families, relations between founders and investors can be complicated.)

The best way to find angel investors is through personal introductions. You could try to cold-call angel groups near you, but angels, like VCs, will pay more attention to deals recommended by someone they respect.

Deal terms with angels vary a lot. There are no generally accepted standards. Sometimes angels' deal terms are as fearsome as VCs'. Other angels, particularly in the earliest stages, will invest based on a two-page agreement.

Angels who only invest occasionally may not themselves know what terms they want. They just want to invest in this startup. What kind of anti-dilution protection do they want? Hell if they know. In these situations, the deal terms tend to be random: the angel asks his lawyer to create a vanilla agreement, and the terms end up being whatever the lawyer considers vanilla. Which in practice usually means, whatever existing agreement he finds lying around his firm. (Few legal documents are created from scratch.)

These heaps o' boilerplate are a problem for small startups, because they tend to grow into the union of all preceding documents. I know of one startup that got from an angel investor what amounted to a five hundred pound handshake: after deciding to invest, the angel presented them with a 70-page agreement. The startup didn't have enough money to pay a lawyer even to read it, let alone negotiate the terms, so the deal fell through.

One solution to this problem would be to have the startup's lawyer produce the agreement, instead of the angel's. Some angels might balk at this, but others would probably welcome it.

Inexperienced angels often get cold feet when the time comes to write that big check. In our startup, one of the two angels in the initial round took months to pay us, and only did after repeated nagging from our lawyer, who was also, fortunately, his lawyer.

It's obvious why investors delay. Investing in startups is risky! When a company is only two months old, every *day* you wait gives you 1.7% more data about their trajectory. But the investor is already being compensated for that risk in the low price of the stock, so it is unfair to delay.

Fair or not, investors do it if you let them. Even VCs do it. And funding delays are a big distraction for founders, who ought to be working on their company, not

worrying about investors. What's a startup to do? With both investors and acquirers, the only leverage you have is competition. If an investor knows you have other investors lined up, he'll be a lot more eager to close-- and not just because he'll worry about losing the deal, but because if other investors are interested, you must be worth investing in. It's the same with acquisitions. No one wants to buy you till someone else wants to buy you, and then everyone wants to buy you.

The key to closing deals is never to stop pursuing alternatives. When an investor says he wants to invest in you, or an acquirer says they want to buy you, *don't believe it till you get the check.* Your natural tendency when an investor says yes will be to relax and go back to writing code. Alas, you can't; you have to keep looking for more investors, if only to get this one to act. [4]

Seed Funding Firms

Seed firms are like angels in that they invest relatively small amounts at early stages, but like VCs in that they're companies that do it as a business, rather than individuals making occasional investments on the side.

Till now, nearly all seed firms have been so-called "incubators," so Y Combinator gets called one too, though the only thing we have in common is that we invest in the earliest phase.

According to the National Association of Business Incubators, there are about 800 incubators in the US. This is an astounding number, because I know the founders of a lot of startups, and I can't think of one that began in an incubator.

What is an incubator? I'm not sure myself. The defining quality seems to be that you work in their space. That's where the name "incubator" comes from. They seem to vary a great deal in other respects. At one extreme is the sort of pork-barrel project where a town gets money from the state government to renovate a vacant building as a "high-tech incubator," as if it were merely lack of the right sort of office space that had till now prevented the town from becoming a startup hub. At the other extreme are places like Idealab, which generates ideas for new startups internally and hires people to work for them.

The classic Bubble incubators, most of which now seem to be dead, were like VC firms except that they took a much bigger role in the startups they funded. In addition to working in their space, you were supposed to use their office staff, lawyers, accountants, and so on.

Whereas incubators tend (or tended) to exert more control than VCs, Y Combinator exerts less. And we think it's better if startups operate out of their own premises, however crappy, than the offices of their investors. So it's annoying that we keep getting called an "incubator," but perhaps inevitable, because there's only one of us so far and no word yet for what we are. If we have to be called something, the obvious name would be "excubator." (The name is more excusable if one considers

it as meaning that we enable people to escape cubicles.)

Because seed firms are companies rather than individual people, reaching them is easier than reaching angels. Just go to their web site and send them an email. The importance of personal introductions varies, but is less than with angels or VCs.

The fact that seed firms are companies also means the investment process is more standardized. (This is generally true with angel groups too.) Seed firms will probably have set deal terms they use for every startup they fund. The fact that the deal terms are standard doesn't mean they're favorable to you, but if other startups have signed the same agreements and things went well for them, it's a sign the terms are reasonable.

Seed firms differ from angels and VCs in that they invest exclusively in the earliest phases—often when the company is still just an idea. Angels and even VC firms occasionally do this, but they also invest at later stages.

The problems are different in the early stages. For example, in the first couple months a startup may completely redefine their idea. So seed investors usually care less about the idea than the people. This is true of all venture funding, but especially so in the seed stage.

Like VCs, one of the advantages of seed firms is the advice they offer. But because seed firms operate in an earlier phase, they need to offer different kinds of advice. For example, a seed firm should be able to give advice about how to approach VCs, which VCs obviously don't need to do; whereas VCs should be able to give advice about how to hire an "executive team," which is not an issue in the seed stage.

In the earliest phases, a lot of the problems are technical, so seed firms should be able to help with technical as well as business problems.

Seed firms and angel investors generally want to invest in the initial phases of a startup, then hand them off to VC firms for the next round. Occasionally startups go from seed funding direct to acquisition, however, and I expect this to become increasingly common.

Google has been aggressively pursuing this route, and now Yahoo is too. Both now compete directly with VCs. And this is a smart move. Why wait for further funding rounds to jack up a startup's price? When a startup reaches the point where VCs have enough information to invest in it, the acquirer should have enough information to buy it. More information, in fact; with their technical depth, the acquirers should be better at picking winners than VCs.

Venture Capital Funds

VC firms are like seed firms in that they're actual companies, but they invest other people's money, and much larger amounts of it. VC investments average several million dollars. So they tend to come later in the life of a startup, are harder to get,

and come with tougher terms.

The word "venture capitalist" is sometimes used loosely for any venture investor, but there is a sharp difference between VCs and other investors: VC firms are organized as *funds*, much like hedge funds or mutual funds. The fund managers, who are called "general partners," get about 2% of the fund annually as a management fee, plus about 20% of the fund's gains.

There is a very sharp dropoff in performance among VC firms, because in the VC business both success and failure are self-perpetuating. When an investment scores spectacularly, as Google did for Kleiner and Sequoia, it generates a lot of good publicity for the VCs. And many founders prefer to take money from successful VC firms, because of the legitimacy it confers. Hence a vicious (for the losers) cycle: VC firms that have been doing badly will only get the deals the bigger fish have rejected, causing them to continue to do badly.

As a result, of the thousand or so VC funds in the US now, only about 50 are likely to make money, and it is very hard for a new fund to break into this group.

In a sense, the lower-tier VC firms are a bargain for founders. They may not be quite as smart or as well connected as the big-name firms, but they are much hungrier for deals. This means you should be able to get better terms from them.

Better how? The most obvious is valuation: they'll take less of your company. But as well as money, there's power. I think founders will increasingly be able to stay on as CEO, and on terms that will make it fairly hard to fire them later.

The most dramatic change, I predict, is that VCs will allow founders to cash out partially by selling some of their stock direct to the VC firm. VCs have traditionally resisted letting founders get anything before the ultimate "liquidity event." But they're also desperate for deals. And since I know from my own experience that the rule against buying stock from founders is a stupid one, this is a natural place for things to give as venture funding becomes more and more a seller's market.

The disadvantage of taking money from less known firms is that people will assume, correctly or not, that you were turned down by the more exalted ones. But, like where you went to college, the name of your VC stops mattering once you have some performance to measure. So the more confident you are, the less you need a brand-name VC. We funded Viaweb entirely with angel money; it never occurred to us that the backing of a well known VC firm would make us seem more impressive. [5]

Another danger of less known firms is that, like angels, they have less reputation to protect. I suspect it's the lower-tier firms that are responsible for most of the tricks that have given VCs such a bad reputation among hackers. They are doubly hosed: the general partners themselves are less able, and yet they have harder problems to solve, because the top VCs skim off all the best deals, leaving the lower-tier firms exactly the startups that are likely to blow up.

For example, lower-tier firms are much more likely to pretend to want to do a deal with you just to lock you up while they decide if they really want to. One experienced CFO said:

> The better ones usually will not give a term sheet unless they really want to do a deal. The second or third tier firms have a much higher break rate—it could be as high as 50%.

It's obvious why: the lower-tier firms' biggest fear, when chance throws them a bone, is that one of the big dogs will notice and take it away. The big dogs don't have to worry about that.

Falling victim to this trick could really hurt you. As one VC told me:

> If you were talking to four VCs, told three of them that you accepted a term sheet, and then have to call them back to tell them you were just kidding, you are absolutely damaged goods.

Here's a partial solution: when a VC offers you a term sheet, ask how many of their last 10 term sheets turned into deals. This will at least force them to lie outright if they want to mislead you.

Not all the people who work at VC firms are partners. Most firms also have a handful of junior employees called something like associates or analysts. If you get a call from a VC firm, go to their web site and check whether the person you talked to is a partner. Odds are it will be a junior person; they scour the web looking for startups their bosses could invest in. The junior people will tend to seem very positive about your company. They're not pretending; they *want* to believe you're a hot prospect, because it would be a huge coup for them if their firm invested in a company they discovered. Don't be misled by this optimism. It's the partners who decide, and they view things with a colder eye.

Because VCs invest large amounts, the money comes with more restrictions. Most only come into effect if the company gets into trouble. For example, VCs generally write it into the deal that in any sale, they get their investment back first. So if the company gets sold at a low price, the founders could get nothing. Some VCs now require that in any sale they get 4x their investment back before the common stock holders (that is, you) get anything, but this is an abuse that should be resisted.

Another difference with large investments is that the founders are usually required to accept "vesting"—to surrender their stock and earn it back over the next 4-5 years. VCs don't want to invest millions in a company the founders could just walk away from. Financially, vesting has little effect, but in some situations it could mean founders will have less power. If VCs got de facto control of the company and fired one of the founders, he'd lose any unvested stock unless there was specific protection against this. So vesting would in that situation force founders to toe the line.

The most noticeable change when a startup takes serious funding is that the founders will no longer have complete control. Ten years ago VCs used to insist that founders step down as CEO and hand the job over to a business guy they supplied. This is less the rule now, partly because the disasters of the Bubble showed that generic business guys don't make such great CEOs.

But while founders will increasingly be able to stay on as CEO, they'll have to cede some power, because the board of directors will become more powerful. In the seed stage, the board is generally a formality; if you want to talk to the other board members, you just yell into the next room. This stops with VC-scale money. In a typical VC funding deal, the board of directors might be composed of two VCs, two founders, and one outside person acceptable to both. The board will have ultimate power, which means the founders now have to convince instead of commanding.

This is not as bad as it sounds, however. Bill Gates is in the same position; he doesn't have majority control of Microsoft; in principle he also has to convince instead of commanding. And yet he seems pretty commanding, doesn't he? As long as things are going smoothly, boards don't interfere much. The danger comes when there's a bump in the road, as happened to Steve Jobs at Apple.

Like angels, VCs prefer to invest in deals that come to them through people they know. So while nearly all VC funds have some address you can send your business plan to, VCs privately admit the chance of getting funding by this route is near zero. One recently told me that he did not know a single startup that got funded this way.

I suspect VCs accept business plans "over the transom" more as a way to keep tabs on industry trends than as a source of deals. In fact, I would strongly advise against mailing your business plan randomly to VCs, because they treat this as evidence of laziness. Do the extra work of getting personal introductions. As one VC put it:

> I'm not hard to find. I know a lot of people. If you can't find some way to reach me, how are you going to create a successful company?

One of the most difficult problems for startup founders is deciding when to approach VCs. You really only get one chance, because they rely heavily on first impressions. And you can't approach some and save others for later, because (a) they ask who else you've talked to and when and (b) they talk among themselves. If you're talking to one VC and he finds out that you were rejected by another several months ago, you'll definitely seem shopworn.

So when do you approach VCs? When you can convince them. If the founders have impressive resumes and the idea isn't hard to understand, you could approach VCs quite early. Whereas if the founders are unknown and the idea is very novel, you might have to launch the thing and show that users loved it before VCs would be

convinced.

If several VCs are interested in you, they will sometimes be willing to split the deal between them. They're more likely to do this if they're close in the VC pecking order. Such deals may be a net win for founders, because you get multiple VCs interested in your success, and you can ask each for advice about the other. One founder I know wrote:

> Two-firm deals are great. It costs you a little more equity, but being able to play the two firms off each other (as well as ask one if the other is being out of line) is invaluable.

When you do negotiate with VCs, remember that they've done this a lot more than you have. They've invested in dozens of startups, whereas this is probably the first you've founded. But don't let them or the situation intimidate you. The average founder is smarter than the average VC. So just do what you'd do in any complex, unfamiliar situation: proceed deliberately, and question anything that seems odd.

It is, unfortunately, common for VCs to put terms in an agreement whose consequences surprise founders later, and also common for VCs to defend things they do by saying that they're standard in the industry. Standard, schmandard; the whole industry is only a few decades old, and rapidly evolving. The concept of "standard" is a useful one when you're operating on a small scale (Y Combinator uses identical terms for every deal because for tiny seed-stage investments it's not worth the overhead of negotiating individual deals), but it doesn't apply at the VC level. On that scale, every negotiation is unique.

Most successful startups get money from more than one of the preceding five sources. [6] And, confusingly, the names of funding sources also tend to be used as the names of different rounds. The best way to explain how it all works is to follow the case of a hypothetical startup.

Stage 1: Seed Round

Our startup begins when a group of three friends have an idea-- either an idea for something they might build, or simply the idea "let's start a company." Presumably they already have some source of food and shelter. But if you have food and shelter, you probably also have something you're supposed to be working on: either classwork, or a job. So if you want to work full-time on a startup, your money situation will probably change too.

A lot of startup founders say they started the company without any idea of what they planned to do. This is actually less common than it seems: many have to claim they thought of the idea after quitting because otherwise their former employer would own it.

The three friends decide to take the leap. Since most startups are in competitive businesses, you not only want to work full-time on them, but more than full-time.

So some or all of the friends quit their jobs or leave school. (Some of the founders in a startup can stay in grad school, but at least one has to make the company his full-time job.)

They're going to run the company out of one of their apartments at first, and since they don't have any users they don't have to pay much for infrastructure. Their main expenses are setting up the company, which costs a couple thousand dollars in legal work and registration fees, and the living expenses of the founders.

The phrase "seed investment" covers a broad range. To some VC firms it means $500,000, but to most startups it means several months' living expenses. We'll suppose our group of friends start with $15,000 from their friend's rich uncle, who they give 5% of the company in return. There's only common stock at this stage. They leave 20% as an options pool for later employees (but they set things up so that they can issue this stock to themselves if they get bought early and most is still unissued), and the three founders each get 25%.

By living really cheaply they think they can make the remaining money last five months. When you have five months' runway left, how soon do you need to start looking for your next round? Answer: immediately. It takes time to find investors, and time (always more than you expect) for the deal to close even after they say yes. So if our group of founders know what they're doing they'll start sniffing around for angel investors right away. But of course their main job is to build version 1 of their software.

The friends might have liked to have more money in this first phase, but being slightly underfunded teaches them an important lesson. For a startup, cheapness is power. The lower your costs, the more options you have—not just at this stage, but at every point till you're profitable. When you have a high "burn rate," you're always under time pressure, which means (a) you don't have time for your ideas to evolve, and (b) you're often forced to take deals you don't like.

Every startup's rule should be: spend little, and work fast.

After ten weeks' work the three friends have built a prototype that gives one a taste of what their product will do. It's not what they originally set out to do—in the process of writing it, they had some new ideas. And it only does a fraction of what the finished product will do, but that fraction includes stuff that no one else has done before.

They've also written at least a skeleton business plan, addressing the five fundamental questions: what they're going to do, why users need it, how large the market is, how they'll make money, and who the competitors are and why this company is going to beat them. (That last has to be more specific than "they suck" or "we'll work really hard.")

If you have to choose between spending time on the demo or the business plan, spend most on the demo. Software is not only more convincing, but a better way

to explore ideas.

Stage 2: Angel Round

While writing the prototype, the group has been traversing their network of friends in search of angel investors. They find some just as the prototype is demoable. When they demo it, one of the angels is willing to invest. Now the group is looking for more money: they want enough to last for a year, and maybe to hire a couple friends. So they're going to raise $200,000.

The angel agrees to invest at a pre-money valuation of $1 million. The company issues $200,000 worth of new shares to the angel; if there were 1000 shares before the deal, this means 200 additional shares. The angel now owns 200/1200 shares, or a sixth of the company, and all the previous shareholders' percentage ownership is diluted by a sixth. After the deal, the capitalization table looks like this:

shareholder	shares	percent
angel	200	16.7
uncle	50	4.2
each founder	250	20.8
option pool	200	16.7
total	1200	100

To keep things simple, I had the angel do a straight cash for stock deal. In reality the angel might be more likely to make the investment in the form of a convertible loan. A convertible loan is a loan that can be converted into stock later; it works out the same as a stock purchase in the end, but gives the angel more protection against being squashed by VCs in future rounds.

Who pays the legal bills for this deal? The startup, remember, only has a couple thousand left. In practice this turns out to be a sticky problem that usually gets solved in some improvised way. Maybe the startup can find lawyers who will do it cheaply in the hope of future work if the startup succeeds. Maybe someone has a lawyer friend. Maybe the angel pays for his lawyer to represent both sides. (Make sure if you take the latter route that the lawyer is *representing* you rather than merely advising you, or his only duty is to the investor.)

An angel investing $200k would probably expect a seat on the board of directors. He might also want preferred stock, meaning a special class of stock that has some additional rights over the common stock everyone else has. Typically these rights include vetoes over major strategic decisions, protection against being diluted in future rounds, and the right to get one's investment back first if the company is sold.

Some investors might expect the founders to accept vesting for a sum this size, and others wouldn't. VCs are more likely to require vesting than angels. At Viaweb

we managed to raise $2.5 million from angels without ever accepting vesting, largely because we were so inexperienced that we were appalled at the idea. In practice this turned out to be good, because it made us harder to push around.

Our experience was unusual; vesting is the norm for amounts that size. Y Combinator doesn't require vesting, because (a) we invest such small amounts, and (b) we think it's unnecessary, and that the hope of getting rich is enough motivation to keep founders at work. But maybe if we were investing millions we would think differently.

I should add that vesting is also a way for founders to protect themselves against one another. It solves the problem of what to do if one of the founders quits. So some founders impose it on themselves when they start the company.

The angel deal takes two weeks to close, so we are now three months into the life of the company.

The point after you get the first big chunk of angel money will usually be the happiest phase in a startup's life. It's a lot like being a postdoc: you have no immediate financial worries, and few responsibilities. You get to work on juicy kinds of work, like designing software. You don't have to spend time on bureaucratic stuff, because you haven't hired any bureaucrats yet. Enjoy it while it lasts, and get as much done as you can, because you will never again be so productive.

With an apparently inexhaustible sum of money sitting safely in the bank, the founders happily set to work turning their prototype into something they can release. They hire one of their friends—at first just as a consultant, so they can try him out—and then a month later as employee #1. They pay him the smallest salary he can live on, plus 3% of the company in restricted stock, vesting over four years. (So after this the option pool is down to 13.7%). [7] They also spend a little money on a freelance graphic designer.

How much stock do you give early employees? That varies so much that there's no conventional number. If you get someone really good, really early, it might be wise to give him as much stock as the founders. The one universal rule is that the amount of stock an employee gets decreases polynomially with the age of the company. In other words, you get rich as a power of how early you were. So if some friends want you to come work for their startup, don't wait several months before deciding.

A month later, at the end of month four, our group of founders have something they can launch. Gradually through word of mouth they start to get users. Seeing the system in use by real users—people they don't know—gives them lots of new ideas. Also they find they now worry obsessively about the status of their server. (How relaxing founders' lives must have been when startups wrote VisiCalc.)

By the end of month six, the system is starting to have a solid core of features,

and a small but devoted following. People start to write about it, and the founders are starting to feel like experts in their field.

We'll assume that their startup is one that could put millions more to use. Perhaps they need to spend a lot on marketing, or build some kind of expensive infrastructure, or hire highly paid salesmen. So they decide to start talking to VCs. They get introductions to VCs from various sources: their angel investor connects them with a couple; they meet a few at conferences; a couple VCs call them after reading about them.

Step 3: Series A Round

Armed with their now somewhat fleshed-out business plan and able to demo a real, working system, the founders visit the VCs they have introductions to. They find the VCs intimidating and inscrutable. They all ask the same question: who else have you pitched to? (VCs are like high school girls: they're acutely aware of their position in the VC pecking order, and their interest in a company is a function of the interest other VCs show in it.)

One of the VC firms says they want to invest and offers the founders a term sheet. A term sheet is a summary of what the deal terms will be when and if they do a deal; lawyers will fill in the details later. By accepting the term sheet, the startup agrees to turn away other VCs for some set amount of time while this firm does the "due diligence" required for the deal. Due diligence is the corporate equivalent of a background check: the purpose is to uncover any hidden bombs that might sink the company later, like serious design flaws in the product, pending lawsuits against the company, intellectual property issues, and so on. VCs' legal and financial due diligence is pretty thorough, but the technical due diligence is generally a joke. [8]

The due diligence discloses no ticking bombs, and six weeks later they go ahead with the deal. Here are the terms: a $2 million investment at a pre-money valuation of $4 million, meaning that after the deal closes the VCs will own a third of the company (2 / (4 + 2)). The VCs also insist that prior to the deal the option pool be enlarged by an additional hundred shares. So the total number of new shares issued is 750, and the cap table becomes:

shareholder	shares	percent	
VCs	650	33.3	
angel	200	10.3	
uncle	50	2.6	
each founder	250	12.8	
employee	36*	1.8	*unvested
option pool	264	13.5	
total	1950	100	

This picture is unrealistic in several respects. For example, while the percentages might end up looking like this, it's unlikely that the VCs would keep the existing

numbers of shares. In fact, every bit of the startup's paperwork would probably be replaced, as if the company were being founded anew. Also, the money might come in several tranches, the later ones subject to various conditions—though this is apparently more common in deals with lower-tier VCs (whose lot in life is to fund more dubious startups) than with the top firms.

And of course any VCs reading this are probably rolling on the floor laughing at how my hypothetical VCs let the angel keep his 10.3 of the company. I admit, this is the Bambi version; in simplifying the picture, I've also made everyone nicer. In the real world, VCs regard angels the way a jealous husband feels about his wife's previous boyfriends. To them the company didn't exist before they invested in it. [9]

I don't want to give the impression you have to do an angel round before going to VCs. In this example I stretched things out to show multiple sources of funding in action. Some startups could go directly from seed funding to a VC round; several of the companies we've funded have.

The founders are required to vest their shares over four years, and the board is now reconstituted to consist of two VCs, two founders, and a fifth person acceptable to both. The angel investor cheerfully surrenders his board seat.

At this point there is nothing new our startup can teach us about funding—or at least, nothing good. [10] The startup will almost certainly hire more people at this point; those millions must be put to work, after all. The company may do additional funding rounds, presumably at higher valuations. They may if they are extraordinarily fortunate do an IPO, which we should remember is also in principle a round of funding, regardless of its de facto purpose. But that, if not beyond the bounds of possibility, is beyond the scope of this article.

Deals Fall Through

Anyone who's been through a startup will find the preceding portrait to be missing something: disasters. If there's one thing all startups have in common, it's that something is always going wrong. And nowhere more than in matters of funding.

For example, our hypothetical startup never spent more than half of one round before securing the next. That's more ideal than typical. Many startups—even successful ones—come close to running out of money at some point. Terrible things happen to startups when they run out of money, because they're designed for growth, not adversity.

But the most unrealistic thing about the series of deals I've described is that they all closed. In the startup world, closing is not what deals do. What deals do is fall through. If you're starting a startup you would do well to remember that. Birds fly; fish swim; deals fall through.

Why? Partly the reason deals seem to fall through so often is that you lie to

yourself. You want the deal to close, so you start to believe it will. But even correcting for this, startup deals fall through alarmingly often—far more often than, say, deals to buy real estate. The reason is that it's such a risky environment. People about to fund or acquire a startup are prone to wicked cases of buyer's remorse. They don't really grasp the risk they're taking till the deal's about to close. And then they panic. And not just inexperienced angel investors, but big companies too.

So if you're a startup founder wondering why some angel investor isn't returning your phone calls, you can at least take comfort in the thought that the same thing is happening to other deals a hundred times the size.

The example of a startup's history that I've presented is like a skeleton—accurate so far as it goes, but needing to be fleshed out to be a complete picture. To get a complete picture, just add in every possible disaster.

A frightening prospect? In a way. And yet also in a way encouraging. The very uncertainty of startups frightens away almost everyone. People overvalue stability—especially young people, who ironically need it least. And so in starting a startup, as in any really bold undertaking, merely deciding to do it gets you halfway there. On the day of the race, most of the other runners won't show up.

Notes

[1] The aim of such regulations is to protect widows and orphans from crooked investment schemes; people with a million dollars in liquid assets are assumed to be able to protect themselves. The unintended consequence is that the investments that generate the highest returns, like hedge funds, are available only to the rich.

[2] Consulting is where product companies go to die. IBM is the most famous example. So starting as a consulting company is like starting out in the grave and trying to work your way up into the world of the living.

[3] If "near you" doesn't mean the Bay Area, Boston, or Seattle, consider moving. It's not a coincidence you haven't heard of many startups from Philadelphia.

[4] Investors are often compared to sheep. And they are like sheep, but that's a rational response to their situation. Sheep act the way they do for a reason. If all the other sheep head for a certain field, it's probably good grazing. And when a wolf appears, is he going to eat a sheep in the middle of the flock, or one near the edge?

[5] This was partly confidence, and partly simple ignorance. We didn't know

ourselves which VC firms were the impressive ones. We thought software was all that mattered. But that turned out to be the right direction to be naive in: it's much better to overestimate than underestimate the importance of making a good product.

[6] I've omitted one source: government grants. I don't think these are even worth thinking about for the average startup. Governments may mean well when they set up grant programs to encourage startups, but what they give with one hand they take away with the other: the process of applying is inevitably so arduous, and the restrictions on what you can do with the money so burdensome, that it would be easier to take a job to get the money.

You should be especially suspicious of grants whose purpose is some kind of social engineering-- e.g. to encourage more startups to be started in Mississippi. Free money to start a startup in a place where few succeed is hardly free.

Some government agencies run venture funding groups, which make investments rather than giving grants. For example, the CIA runs a venture fund called In-Q-Tel that is modelled on private sector funds and apparently generates good returns. They would probably be worth approaching—if you don't mind taking money from the CIA.

[7] Options have largely been replaced with restricted stock, which amounts to the same thing. Instead of earning the right to buy stock, the employee gets the stock up front, and earns the right not to have to give it back. The shares set aside for this purpose are still called the "option pool."

[8] First-rate technical people do not generally hire themselves out to do due diligence for VCs. So the most difficult part for startup founders is often responding politely to the inane questions of the "expert" they send to look you over.

[9] VCs regularly wipe out angels by issuing arbitrary amounts of new stock. They seem to have a standard piece of casuistry for this situation: that the angels are no longer working to help the company, and so don't deserve to keep their stock. This of course reflects a willful misunderstanding of what investment means; like any investor, the angel is being compensated for risks he took earlier. By a similar logic, one could argue that the VCs should be deprived of their shares when the company goes public.

[10] One new thing the company might encounter is a *down round*, or a funding round at valuation lower than the previous round. Down rounds are bad news; it is generally the common stock holders who take the hit. Some of the most fearsome provisions in VC deal terms have to do with down rounds—like "full ratchet anti-dilution," which is as frightening as it sounds.

Founders are tempted to ignore these clauses, because they think the company will either be a big success or a complete bust. VCs know otherwise: it's not uncommon for startups to have moments of adversity before they ultimately

succeed. So it's worth negotiating anti-dilution provisions, even though you don't think you need to, and VCs will try to make you feel that you're being gratuitously troublesome.

Thanks to Sam Altman, Hutch Fishman, Steve Huffman, Jessica Livingston, Sesha Pratap, Stan Reiss, Andy Singleton, Zak Stone, and Aaron Swartz for reading drafts of this.

Arabic Translation

Web 2.0

November 2005

Does "Web 2.0" mean anything? Till recently I thought it didn't, but the truth turns out to be more complicated. Originally, yes, it was meaningless. Now it seems to have acquired a meaning. And yet those who dislike the term are probably right, because if it means what I think it does, we don't need it.

I first heard the phrase "Web 2.0" in the name of the Web 2.0 conference in 2004. At the time it was supposed to mean using "the web as a platform," which I took to refer to web-based applications. [1]

So I was surprised at a conference this summer when Tim O'Reilly led a session intended to figure out a definition of "Web 2.0." Didn't it already mean using the web as a platform? And if it didn't already mean something, why did we need the phrase at all?

Origins

Tim says the phrase "Web 2.0" first arose in "a brainstorming session between O'Reilly and Medialive International." What is Medialive International? "Producers of technology tradeshows and conferences," according to their site. So presumably that's what this brainstorming session was about. O'Reilly wanted to organize a conference about the web, and they were wondering what to call it.

I don't think there was any deliberate plan to suggest there was a new *version* of the web. They just wanted to make the point that the web mattered again. It was a kind of semantic deficit spending: they knew new things were coming, and the "2.0" referred to whatever those might turn out to be.

And they were right. New things were coming. But the new version number led to some awkwardness in the short term. In the process of developing the pitch for the first conference, someone must have decided they'd better take a stab at explaining what that "2.0" referred to. Whatever it meant, "the web as a platform" was at least not too constricting.

The story about "Web 2.0" meaning the web as a platform didn't live much past the first conference. By the second conference, what "Web 2.0" seemed to mean was something about democracy. At least, it did when people wrote about it online. The conference itself didn't seem very grassroots. It cost $2800, so the only people who could afford to go were VCs and people from big companies.

And yet, oddly enough, Ryan Singel's article about the conference in *Wired News*

spoke of "throngs of geeks." When a friend of mine asked Ryan about this, it was news to him. He said he'd originally written something like "throngs of VCs and biz dev guys" but had later shortened it just to "throngs," and that this must have in turn been expanded by the editors into "throngs of geeks." After all, a Web 2.0 conference would presumably be full of geeks, right?

Well, no. There were about 7. Even Tim O'Reilly was wearing a suit, a sight so alien I couldn't parse it at first. I saw him walk by and said to one of the O'Reilly people "that guy looks just like Tim."

"Oh, that's Tim. He bought a suit." I ran after him, and sure enough, it was. He explained that he'd just bought it in Thailand.

The 2005 Web 2.0 conference reminded me of Internet trade shows during the Bubble, full of prowling VCs looking for the next hot startup. There was that same odd atmosphere created by a large number of people determined not to miss out. Miss out on what? They didn't know. Whatever was going to happen—whatever Web 2.0 turned out to be.

I wouldn't quite call it "Bubble 2.0" just because VCs are eager to invest again. The Internet is a genuinely big deal. The bust was as much an overreaction as the boom. It's to be expected that once we started to pull out of the bust, there would be a lot of growth in this area, just as there was in the industries that spiked the sharpest before the Depression.

The reason this won't turn into a second Bubble is that the IPO market is gone. Venture investors are driven by exit strategies. The reason they were funding all those laughable startups during the late 90s was that they hoped to sell them to gullible retail investors; they hoped to be laughing all the way to the bank. Now that route is closed. Now the default exit strategy is to get bought, and acquirers are less prone to irrational exuberance than IPO investors. The closest you'll get to Bubble valuations is Rupert Murdoch paying $580 million for Myspace. That's only off by a factor of 10 or so.

1. Ajax

Does "Web 2.0" mean anything more than the name of a conference yet? I don't like to admit it, but it's starting to. When people say "Web 2.0" now, I have some idea what they mean. And the fact that I both despise the phrase and understand it is the surest proof that it has started to mean something.

One ingredient of its meaning is certainly Ajax, which I can still only just bear to use without scare quotes. Basically, what "Ajax" means is "Javascript now works." And that in turn means that web-based applications can now be made to work much more like desktop ones.

As you read this, a whole new generation of software is being written to take advantage of Ajax. There hasn't been such a wave of new applications since microcomputers first appeared. Even Microsoft sees it, but it's too late for them to do anything more than leak "internal" documents designed to give the impression they're on top of this new trend.

In fact the new generation of software is being written way too fast for Microsoft even to channel it, let alone write their own in house. Their only hope now is to

buy all the best Ajax startups before Google does. And even that's going to be hard, because Google has as big a head start in buying microstartups as it did in search a few years ago. After all, Google Maps, the canonical Ajax application, was the result of a startup they bought.

So ironically the original description of the Web 2.0 conference turned out to be partially right: web-based applications are a big component of Web 2.0. But I'm convinced they got this right by accident. The Ajax boom didn't start till early 2005, when Google Maps appeared and the term "Ajax" was coined.

2. Democracy

The second big element of Web 2.0 is democracy. We now have several examples to prove that amateurs can surpass professionals, when they have the right kind of system to channel their efforts. Wikipedia may be the most famous. Experts have given Wikipedia middling reviews, but they miss the critical point: it's good enough. And it's free, which means people actually read it. On the web, articles you have to pay for might as well not exist. Even if you were willing to pay to read them yourself, you can't link to them. They're not part of the conversation.

Another place democracy seems to win is in deciding what counts as news. I never look at any news site now except Reddit. [2] I know if something major happens, or someone writes a particularly interesting article, it will show up there. Why bother checking the front page of any specific paper or magazine? Reddit's like an RSS feed for the whole web, with a filter for quality. Similar sites include Digg, a technology news site that's rapidly approaching Slashdot in popularity, and del.icio.us, the collaborative bookmarking network that set off the "tagging" movement. And whereas Wikipedia's main appeal is that it's good enough and free, these sites suggest that voters do a significantly better job than human editors.

The most dramatic example of Web 2.0 democracy is not in the selection of ideas, but their production. I've noticed for a while that the stuff I read on individual people's sites is as good as or better than the stuff I read in newspapers and magazines. And now I have independent evidence: the top links on Reddit are generally links to individual people's sites rather than to magazine articles or news stories.

My experience of writing for magazines suggests an explanation. Editors. They control the topics you can write about, and they can generally rewrite whatever you produce. The result is to damp extremes. Editing yields 95th percentile writing —95% of articles are improved by it, but 5% are dragged down. 5% of the time you get "throngs of geeks."

On the web, people can publish whatever they want. Nearly all of it falls short of the editor-damped writing in print publications. But the pool of writers is very, very large. If it's large enough, the lack of damping means the best writing online should surpass the best in print. [3] And now that the web has evolved mechanisms for selecting good stuff, the web wins net. Selection beats damping, for the same reason market economies beat centrally planned ones.

Even the startups are different this time around. They are to the startups of the Bubble what bloggers are to the print media. During the Bubble, a startup meant a company headed by an MBA that was blowing through several million dollars of VC money to "get big fast" in the most literal sense. Now it means a smaller, younger,

more technical group that just decided to make something great. They'll decide later if they want to raise VC-scale funding, and if they take it, they'll take it on their terms.

3. Don't Maltreat Users

I think everyone would agree that democracy and Ajax are elements of "Web 2.0." I also see a third: not to maltreat users. During the Bubble a lot of popular sites were quite high-handed with users. And not just in obvious ways, like making them register, or subjecting them to annoying ads. The very design of the average site in the late 90s was an abuse. Many of the most popular sites were loaded with obtrusive branding that made them slow to load and sent the user the message: this is our site, not yours. (There's a physical analog in the Intel and Microsoft stickers that come on some laptops.)

I think the root of the problem was that sites felt they were giving something away for free, and till recently a company giving anything away for free could be pretty high-handed about it. Sometimes it reached the point of economic sadism: site owners assumed that the more pain they caused the user, the more benefit it must be to them. The most dramatic remnant of this model may be at salon.com, where you can read the beginning of a story, but to get the rest you have sit through a *movie*.

At Y Combinator we advise all the startups we fund never to lord it over users. Never make users register, unless you need to in order to store something for them. If you do make users register, never make them wait for a confirmation link in an email; in fact, don't even ask for their email address unless you need it for some reason. Don't ask them any unnecessary questions. Never send them email unless they explicitly ask for it. Never frame pages you link to, or open them in new windows. If you have a free version and a pay version, don't make the free version too restricted. And if you find yourself asking "should we allow users to do x?" just answer "yes" whenever you're unsure. Err on the side of generosity.

In How to Start a Startup I advised startups never to let anyone fly under them, meaning never to let any other company offer a cheaper, easier solution. Another way to fly low is to give users more power. Let users do what they want. If you don't and a competitor does, you're in trouble.

iTunes is Web 2.0ish in this sense. Finally you can buy individual songs instead of having to buy whole albums. The recording industry hated the idea and resisted it as long as possible. But it was obvious what users wanted, so Apple flew under the labels. [4] Though really it might be better to describe iTunes as Web 1.5. Web 2.0 applied to music would probably mean individual bands giving away DRMless songs for free.

The ultimate way to be nice to users is to give them something for free that competitors charge for. During the 90s a lot of people probably thought we'd have some working system for micropayments by now. In fact things have gone in the other direction. The most successful sites are the ones that figure out new ways to give stuff away for free. Craigslist has largely destroyed the classified ad sites of the 90s, and OkCupid looks likely to do the same to the previous generation of dating sites.

Serving web pages is very, very cheap. If you can make even a fraction of a cent

per page view, you can make a profit. And technology for targeting ads continues to improve. I wouldn't be surprised if ten years from now eBay had been supplanted by an ad-supported freeBay (or, more likely, gBay).

Odd as it might sound, we tell startups that they should try to make as little money as possible. If you can figure out a way to turn a billion dollar industry into a fifty million dollar industry, so much the better, if all fifty million go to you. Though indeed, making things cheaper often turns out to generate more money in the end, just as automating things often turns out to generate more jobs.

The ultimate target is Microsoft. What a bang that balloon is going to make when someone pops it by offering a free web-based alternative to MS Office. [5] Who will? Google? They seem to be taking their time. I suspect the pin will be wielded by a couple of 20 year old hackers who are too naive to be intimidated by the idea. (How hard can it be?)

The Common Thread

Ajax, democracy, and not dissing users. What do they all have in common? I didn't realize they had anything in common till recently, which is one of the reasons I disliked the term "Web 2.0" so much. It seemed that it was being used as a label for whatever happened to be new—that it didn't predict anything.

But there is a common thread. Web 2.0 means using the web the way it's meant to be used. The "trends" we're seeing now are simply the inherent nature of the web emerging from under the broken models that got imposed on it during the Bubble.

I realized this when I read an interview with Joe Kraus, the co-founder of Excite. [6]

> Excite really never got the business model right at all. We fell into the classic problem of how when a new medium comes out it adopts the practices, the content, the business models of the old medium—which fails, and then the more appropriate models get figured out.

It may have seemed as if not much was happening during the years after the Bubble burst. But in retrospect, something was happening: the web was finding its natural angle of repose. The democracy component, for example—that's not an innovation, in the sense of something someone made happen. That's what the web naturally tends to produce.

Ditto for the idea of delivering desktop-like applications over the web. That idea is almost as old as the web. But the first time around it was co-opted by Sun, and we got Java applets. Java has since been remade into a generic replacement for C++, but in 1996 the story about Java was that it represented a new model of software. Instead of desktop applications, you'd run Java "applets" delivered from a server.

This plan collapsed under its own weight. Microsoft helped kill it, but it would have died anyway. There was no uptake among hackers. When you find PR firms promoting something as the next development platform, you can be sure it's not. If it were, you wouldn't need PR firms to tell you, because hackers would already be writing stuff on top of it, the way sites like Busmonster used Google Maps as a

platform before Google even meant it to be one.

The proof that Ajax is the next hot platform is that thousands of hackers have spontaneously started building things on top of it. Mikey likes it.

There's another thing all three components of Web 2.0 have in common. Here's a clue. Suppose you approached investors with the following idea for a Web 2.0 startup:

> Sites like del.icio.us and flickr allow users to "tag" content with descriptive tokens. But there is also huge source of *implicit* tags that they ignore: the text within web links. Moreover, these links represent a social network connecting the individuals and organizations who created the pages, and by using graph theory we can compute from this network an estimate of the reputation of each member. We plan to mine the web for these implicit tags, and use them together with the reputation hierarchy they embody to enhance web searches.

How long do you think it would take them on average to realize that it was a description of Google?

Google was a pioneer in all three components of Web 2.0: their core business sounds crushingly hip when described in Web 2.0 terms, "Don't maltreat users" is a subset of "Don't be evil," and of course Google set off the whole Ajax boom with Google Maps.

Web 2.0 means using the web as it was meant to be used, and Google does. That's their secret. They're sailing with the wind, instead of sitting becalmed praying for a business model, like the print media, or trying to tack upwind by suing their customers, like Microsoft and the record labels. [7]

Google doesn't try to force things to happen their way. They try to figure out what's going to happen, and arrange to be standing there when it does. That's the way to approach technology—and as business includes an ever larger technological component, the right way to do business.

The fact that Google is a "Web 2.0" company shows that, while meaningful, the term is also rather bogus. It's like the word "allopathic." It just means doing things right, and it's a bad sign when you have a special word for that.

Notes

[1] From the conference site, June 2004: "While the first wave of the Web was closely tied to the browser, the second wave extends applications across the web and enables a new generation of services and business opportunities." To the extent this means anything, it seems to be about web-based applications.

[2] Disclosure: Reddit was funded by Y Combinator. But although I started using it out of loyalty to the home team, I've become a genuine addict. While we're at it, I'm also an investor in !MSFT, having sold all my shares earlier this year.

[3] I'm not against editing. I spend more time editing than writing, and I have a group of picky friends who proofread almost everything I write. What I dislike is editing done after the fact by someone else.

[4] Obvious is an understatement. Users had been climbing in through the window for years before Apple finally moved the door.

[5] Hint: the way to create a web-based alternative to Office may not be to write every component yourself, but to establish a protocol for web-based apps to share a virtual home directory spread across multiple servers. Or it may be to write it all yourself.

[6] In Jessica Livingston's *Founders at Work*.

[7] Microsoft didn't sue their customers directly, but they seem to have done all they could to help SCO sue them.

Thanks to Trevor Blackwell, Sarah Harlin, Jessica Livingston, Peter Norvig, Aaron Swartz, and Jeff Weiner for reading drafts of this, and to the guys at O'Reilly and Adaptive Path for answering my questions.

Interview About Web 2.0

Spanish Translation

German Translation

Russian Translation

Japanese Translation

If you liked this, you may also like ***Hackers & Painters***.

Good and Bad Procrastination

December 2005

The most impressive people I know are all terrible procrastinators. So could it be that procrastination isn't always bad?

Most people who write about procrastination write about how to cure it. But this is, strictly speaking, impossible. There are an infinite number of things you could be doing. No matter what you work on, you're not working on everything else. So the question is not how to avoid procrastination, but how to procrastinate well.

There are three variants of procrastination, depending on what you do instead of working on something: you could work on (a) nothing, (b) something less important, or (c) something more important. That last type, I'd argue, is good procrastination.

That's the "absent-minded professor," who forgets to shave, or eat, or even perhaps look where he's going while he's thinking about some interesting question. His mind is absent from the everyday world because it's hard at work in another.

That's the sense in which the most impressive people I know are all procrastinators. They're type-C procrastinators: they put off working on small stuff to work on big stuff.

What's "small stuff?" Roughly, work that has zero chance of being mentioned in your obituary. It's hard to say at the time what will turn out to be your best work (will it be your magnum opus on Sumerian temple architecture, or the detective thriller you wrote under a pseudonym?), but there's a whole class of tasks you can safely rule out: shaving, doing your laundry, cleaning the house, writing thank-you notes—anything that might be called an errand.

Good procrastination is avoiding errands to do real work.

Good in a sense, at least. The people who want you to do the errands won't think it's good. But you probably have to annoy them if you want to get anything done. The mildest seeming people, if they want to do real work, all have a certain degree of ruthlessness when it comes to avoiding errands.

Some errands, like replying to letters, go away if you ignore them (perhaps taking

friends with them). Others, like mowing the lawn, or filing tax returns, only get worse if you put them off. In principle it shouldn't work to put off the second kind of errand. You're going to have to do whatever it is eventually. Why not (as past-due notices are always saying) do it now?

The reason it pays to put off even those errands is that real work needs two things errands don't: big chunks of time, and the right mood. If you get inspired by some project, it can be a net win to blow off everything you were supposed to do for the next few days to work on it. Yes, those errands may cost you more time when you finally get around to them. But if you get a lot done during those few days, you will be net more productive.

In fact, it may not be a difference in degree, but a difference in kind. There may be types of work that can only be done in long, uninterrupted stretches, when inspiration hits, rather than dutifully in scheduled little slices. Empirically it seems to be so. When I think of the people I know who've done great things, I don't imagine them dutifully crossing items off to-do lists. I imagine them sneaking off to work on some new idea.

Conversely, forcing someone to perform errands synchronously is bound to limit their productivity. The cost of an interruption is not just the time it takes, but that it breaks the time on either side in half. You probably only have to interrupt someone a couple times a day before they're unable to work on hard problems at all.

I've wondered a lot about why startups are most productive at the very beginning, when they're just a couple guys in an apartment. The main reason may be that there's no one to interrupt them yet. In theory it's good when the founders finally get enough money to hire people to do some of the work for them. But it may be better to be overworked than interrupted. Once you dilute a startup with ordinary office workers—with type-B procrastinators—the whole company starts to resonate at their frequency. They're interrupt-driven, and soon you are too.

Errands are so effective at killing great projects that a lot of people use them for that purpose. Someone who has decided to write a novel, for example, will suddenly find that the house needs cleaning. People who fail to write novels don't do it by sitting in front of a blank page for days without writing anything. They do it by feeding the cat, going out to buy something they need for their apartment, meeting a friend for coffee, checking email. "I don't have time to work," they say. And they don't; they've made sure of that.

(There's also a variant where one has no place to work. The cure is to visit the places where famous people worked, and see how unsuitable they were.)

I've used both these excuses at one time or another. I've learned a lot of tricks for making myself work over the last 20 years, but even now I don't win consistently. Some days I get real work done. Other days are eaten up by errands. And I know it's usually my fault: I *let* errands eat up the day, to avoid facing some hard

problem.

The most dangerous form of procrastination is unacknowledged type-B procrastination, because it doesn't feel like procrastination. You're "getting things done." Just the wrong things.

Any advice about procrastination that concentrates on crossing things off your to-do list is not only incomplete, but positively misleading, if it doesn't consider the possibility that the to-do list is itself a form of type-B procrastination. In fact, possibility is too weak a word. Nearly everyone's is. Unless you're working on the biggest things you could be working on, you're type-B procrastinating, no matter how much you're getting done.

In his famous essay You and Your Research (which I recommend to anyone ambitious, no matter what they're working on), Richard Hamming suggests that you ask yourself three questions:

1. What are the most important problems in your field?

2. Are you working on one of them?

3. Why not?

Hamming was at Bell Labs when he started asking such questions. In principle anyone there ought to have been able to work on the most important problems in their field. Perhaps not everyone can make an equally dramatic mark on the world; I don't know; but whatever your capacities, there are projects that stretch them. So Hamming's exercise can be generalized to:

> What's the best thing you could be working on, and why aren't you?

Most people will shy away from this question. I shy away from it myself; I see it there on the page and quickly move on to the next sentence. Hamming used to go around actually asking people this, and it didn't make him popular. But it's a question anyone ambitious should face.

The trouble is, you may end up hooking a very big fish with this bait. To do good work, you need to do more than find good projects. Once you've found them, you have to get yourself to work on them, and that can be hard. The bigger the problem, the harder it is to get yourself to work on it.

Of course, the main reason people find it difficult to work on a particular problem is that they don't enjoy it. When you're young, especially, you often find yourself working on stuff you don't really like-- because it seems impressive, for example, or because you've been assigned to work on it. Most grad students are stuck working on big problems they don't really like, and grad school is thus synonymous with procrastination.

But even when you like what you're working on, it's easier to get yourself to work

on small problems than big ones. Why? Why is it so hard to work on big problems? One reason is that you may not get any reward in the forseeable future. If you work on something you can finish in a day or two, you can expect to have a nice feeling of accomplishment fairly soon. If the reward is indefinitely far in the future, it seems less real.

Another reason people don't work on big projects is, ironically, fear of wasting time. What if they fail? Then all the time they spent on it will be wasted. (In fact it probably won't be, because work on hard projects almost always leads somewhere.)

But the trouble with big problems can't be just that they promise no immediate reward and might cause you to waste a lot of time. If that were all, they'd be no worse than going to visit your in-laws. There's more to it than that. Big problems are *terrifying*. There's an almost physical pain in facing them. It's like having a vacuum cleaner hooked up to your imagination. All your initial ideas get sucked out immediately, and you don't have any more, and yet the vacuum cleaner is still sucking.

You can't look a big problem too directly in the eye. You have to approach it somewhat obliquely. But you have to adjust the angle just right: you have to be facing the big problem directly enough that you catch some of the excitement radiating from it, but not so much that it paralyzes you. You can tighten the angle once you get going, just as a sailboat can sail closer to the wind once it gets underway.

If you want to work on big things, you seem to have to trick yourself into doing it. You have to work on small things that could grow into big things, or work on successively larger things, or split the moral load with collaborators. It's not a sign of weakness to depend on such tricks. The very best work has been done this way.

When I talk to people who've managed to make themselves work on big things, I find that all blow off errands, and all feel guilty about it. I don't think they should feel guilty. There's more to do than anyone could. So someone doing the best work they can is inevitably going to leave a lot of errands undone. It seems a mistake to feel bad about that.

I think the way to "solve" the problem of procrastination is to let delight pull you instead of making a to-do list push you. Work on an ambitious project you really enjoy, and sail as close to the wind as you can, and you'll leave the right things undone.

Thanks to Trevor Blackwell, Jessica Livingston, and Robert Morris for reading

drafts of this.

- [Romanian Translation]

- [Russian Translation]

- [Hebrew Translation]

- [German Translation]

- [Portuguese Translation]

- [Italian Translation]

- [Japanese Translation]

- [Spanish Translation]

How to Do What You Love

January 2006

To do something well you have to like it. That idea is not exactly novel. We've got it down to four words: "Do what you love." But it's not enough just to tell people that. Doing what you love is complicated.

The very idea is foreign to what most of us learn as kids. When I was a kid, it seemed as if work and fun were opposites by definition. Life had two states: some of the time adults were making you do things, and that was called work; the rest of the time you could do what you wanted, and that was called playing. Occasionally the things adults made you do were fun, just as, occasionally, playing wasn't — for example, if you fell and hurt yourself. But except for these few anomalous cases, work was pretty much defined as not-fun.

And it did not seem to be an accident. School, it was implied, was tedious *because* it was preparation for grownup work.

The world then was divided into two groups, grownups and kids. Grownups, like some kind of cursed race, had to work. Kids didn't, but they did have to go to school, which was a dilute version of work meant to prepare us for the real thing. Much as we disliked school, the grownups all agreed that grownup work was worse, and that we had it easy.

Teachers in particular all seemed to believe implicitly that work was not fun. Which is not surprising: work wasn't fun for most of them. Why did we have to memorize state capitals instead of playing dodgeball? For the same reason they had to watch over a bunch of kids instead of lying on a beach. You couldn't just do what you wanted.

I'm not saying we should let little kids do whatever they want. They may have to be made to work on certain things. But if we make kids work on dull stuff, it might be wise to tell them that tediousness is not the defining quality of work, and indeed that the reason they have to work on dull stuff now is so they can work on more interesting stuff later. [1]

Once, when I was about 9 or 10, my father told me I could be whatever I wanted when I grew up, so long as I enjoyed it. I remember that precisely because it seemed so anomalous. It was like being told to use dry water. Whatever I thought he meant, I didn't think he meant work could *literally* be fun — fun like playing. It took me years to grasp that.

Jobs

By high school, the prospect of an actual job was on the horizon. Adults would sometimes come to speak to us about their work, or we would go to see them at work. It was always understood that they enjoyed what they did. In retrospect I think one may have: the private jet pilot. But I don't think the bank manager really did.

The main reason they all acted as if they enjoyed their work was presumably the upper-middle class convention that you're supposed to. It would not merely be bad for your career to say that you despised your job, but a social faux-pas.

Why is it conventional to pretend to like what you do? The first sentence of this essay explains that. If you have to like something to do it well, then the most successful people will all like what they do. That's where the upper-middle class tradition comes from. Just as houses all over America are full of chairs that are, without the owners even knowing it, nth-degree imitations of chairs designed 250 years ago for French kings, conventional attitudes about work are, without the owners even knowing it, nth-degree imitations of the attitudes of people who've done great things.

What a recipe for alienation. By the time they reach an age to think about what they'd like to do, most kids have been thoroughly misled about the idea of loving one's work. School has trained them to regard work as an unpleasant duty. Having a job is said to be even more onerous than schoolwork. And yet all the adults claim to like what they do. You can't blame kids for thinking "I am not like these people; I am not suited to this world."

Actually they've been told three lies: the stuff they've been taught to regard as work in school is not real work; grownup work is not (necessarily) worse than schoolwork; and many of the adults around them are lying when they say they like what they do.

The most dangerous liars can be the kids' own parents. If you take a boring job to give your family a high standard of living, as so many people do, you risk infecting your kids with the idea that work is boring. [2] Maybe it would be better for kids in this one case if parents were not so unselfish. A parent who set an example of loving their work might help their kids more than an expensive house. [3]

It was not till I was in college that the idea of work finally broke free from the idea of making a living. Then the important question became not how to make money, but what to work on. Ideally these coincided, but some spectacular boundary cases (like Einstein in the patent office) proved they weren't identical.

The definition of work was now to make some original contribution to the world, and in the process not to starve. But after the habit of so many years my idea of work still included a large component of pain. Work still seemed to require discipline, because only hard problems yielded grand results, and hard problems couldn't literally be fun. Surely one had to force oneself to work on them.

If you think something's supposed to hurt, you're less likely to notice if you're doing it wrong. That about sums up my experience of graduate school.

Bounds

How much are you supposed to like what you do? Unless you know that, you don't know when to stop searching. And if, like most people, you underestimate it, you'll tend to stop searching too early. You'll end up doing something chosen for you by your parents, or the desire to make money, or prestige — or sheer inertia.

Here's an upper bound: Do what you love doesn't mean, do what you would like to do most *this second*. Even Einstein probably had moments when he wanted to have a cup of coffee, but told himself he ought to finish what he was working on first.

It used to perplex me when I read about people who liked what they did so much that there was nothing they'd rather do. There didn't seem to be any sort of work I liked *that* much. If I had a choice of (a) spending the next hour working on something or (b) be teleported to Rome and spend the next hour wandering about, was there any sort of work I'd prefer? Honestly, no.

But the fact is, almost anyone would rather, at any given moment, float about in the Carribbean, or have sex, or eat some delicious food, than work on hard problems. The rule about doing what you love assumes a certain length of time. It doesn't mean, do what will make you happiest this second, but what will make you happiest over some longer period, like a week or a month.

Unproductive pleasures pall eventually. After a while you get tired of lying on the beach. If you want to stay happy, you have to do something.

As a lower bound, you have to like your work more than any unproductive pleasure. You have to like what you do enough that the concept of "spare time" seems mistaken. Which is not to say you have to spend all your time working. You can only work so much before you get tired and start to screw up. Then you want to do something else — even something mindless. But you don't regard this time as the prize and the time you spend working as the pain you endure to earn it.

I put the lower bound there for practical reasons. If your work is not your favorite thing to do, you'll have terrible problems with procrastination. You'll have to force yourself to work, and when you resort to that the results are distinctly inferior.

To be happy I think you have to be doing something you not only enjoy, but admire. You have to be able to say, at the end, wow, that's pretty cool. This doesn't mean you have to make something. If you learn how to hang glide, or to speak a foreign language fluently, that will be enough to make you say, for a while at least, wow, that's pretty cool. What there has to be is a test.

So one thing that falls just short of the standard, I think, is reading books. Except for some books in math and the hard sciences, there's no test of how well you've read a book, and that's why merely reading books doesn't quite feel like work. You have to do something with what you've read to feel productive.

I think the best test is one Gino Lee taught me: to try to do things that would make your friends say wow. But it probably wouldn't start to work properly till about age 22, because most people haven't had a big enough sample to pick friends from before then.

Sirens

What you should not do, I think, is worry about the opinion of anyone beyond your friends. You shouldn't worry about prestige. Prestige is the opinion of the rest of the world. When you can ask the opinions of people whose judgement you respect, what does it add to consider the opinions of people you don't even know? [4]

This is easy advice to give. It's hard to follow, especially when you're young. [5] Prestige is like a powerful magnet that warps even your beliefs about what you enjoy. It causes you to work not on what you like, but what you'd like to like.

That's what leads people to try to write novels, for example. They like reading novels. They notice that people who write them win Nobel prizes. What could be more wonderful, they think, than to be a novelist? But liking the idea of being a novelist is not enough; you have to like the actual work of novel-writing if you're going to be good at it; you have to like making up elaborate lies.

Prestige is just fossilized inspiration. If you do anything well enough, you'll *make* it prestigious. Plenty of things we now consider prestigious were anything but at first. Jazz comes to mind — though almost any established art form would do. So just do what you like, and let prestige take care of itself.

Prestige is especially dangerous to the ambitious. If you want to make ambitious people waste their time on errands, the way to do it is to bait the hook with prestige. That's the recipe for getting people to give talks, write forewords, serve on committees, be department heads, and so on. It might be a good rule simply to avoid any prestigious task. If it didn't suck, they wouldn't have had to make it prestigious.

Similarly, if you admire two kinds of work equally, but one is more prestigious, you should probably choose the other. Your opinions about what's admirable are always going to be slightly influenced by prestige, so if the two seem equal to you, you probably have more genuine admiration for the less prestigious one.

The other big force leading people astray is money. Money by itself is not that dangerous. When something pays well but is regarded with contempt, like telemarketing, or prostitution, or personal injury litigation, ambitious people aren't tempted by it. That kind of work ends up being done by people who are "just trying to make a living." (Tip: avoid any field whose practitioners say this.) The danger is when money is combined with prestige, as in, say, corporate law, or medicine. A comparatively safe and prosperous career with some automatic baseline prestige is dangerously tempting to someone young, who hasn't thought much about what they really like.

The test of whether people love what they do is whether they'd do it even if they weren't paid for it — even if they had to work at another job to make a living. How many corporate lawyers would do their current work if they had to do it for free, in their spare time, and take day jobs as waiters to support themselves?

This test is especially helpful in deciding between different kinds of academic work, because fields vary greatly in this respect. Most good mathematicians would work on math even if there were no jobs as math professors, whereas in the departments at the other end of the spectrum, the availability of teaching jobs is the driver: people would rather be English professors than work in ad agencies, and publishing papers is the way you compete for such jobs. Math would happen without math departments, but it is the existence of English majors, and therefore

jobs teaching them, that calls into being all those thousands of dreary papers about gender and identity in the novels of Conrad. No one does that kind of thing for fun.

The advice of parents will tend to err on the side of money. It seems safe to say there are more undergrads who want to be novelists and whose parents want them to be doctors than who want to be doctors and whose parents want them to be novelists. The kids think their parents are "materialistic." Not necessarily. All parents tend to be more conservative for their kids than they would for themselves, simply because, as parents, they share risks more than rewards. If your eight year old son decides to climb a tall tree, or your teenage daughter decides to date the local bad boy, you won't get a share in the excitement, but if your son falls, or your daughter gets pregnant, you'll have to deal with the consequences.

Discipline

With such powerful forces leading us astray, it's not surprising we find it so hard to discover what we like to work on. Most people are doomed in childhood by accepting the axiom that work = pain. Those who escape this are nearly all lured onto the rocks by prestige or money. How many even discover something they love to work on? A few hundred thousand, perhaps, out of billions.

It's hard to find work you love; it must be, if so few do. So don't underestimate this task. And don't feel bad if you haven't succeeded yet. In fact, if you admit to yourself that you're discontented, you're a step ahead of most people, who are still in denial. If you're surrounded by colleagues who claim to enjoy work that you find contemptible, odds are they're lying to themselves. Not necessarily, but probably.

Although doing great work takes less discipline than people think — because the way to do great work is to find something you like so much that you don't have to force yourself to do it — *finding* work you love does usually require discipline. Some people are lucky enough to know what they want to do when they're 12, and just glide along as if they were on railroad tracks. But this seems the exception. More often people who do great things have careers with the trajectory of a ping-pong ball. They go to school to study A, drop out and get a job doing B, and then become famous for C after taking it up on the side.

Sometimes jumping from one sort of work to another is a sign of energy, and sometimes it's a sign of laziness. Are you dropping out, or boldly carving a new path? You often can't tell yourself. Plenty of people who will later do great things seem to be disappointments early on, when they're trying to find their niche.

Is there some test you can use to keep yourself honest? One is to try to do a good job at whatever you're doing, even if you don't like it. Then at least you'll know you're not using dissatisfaction as an excuse for being lazy. Perhaps more importantly, you'll get into the habit of doing things well.

Another test you can use is: always produce. For example, if you have a day job you don't take seriously because you plan to be a novelist, are you producing? Are you writing pages of fiction, however bad? As long as you're producing, you'll know you're not merely using the hazy vision of the grand novel you plan to write one day as an opiate. The view of it will be obstructed by the all too palpably flawed one you're actually writing.

"Always produce" is also a heuristic for finding the work you love. If you subject yourself to that constraint, it will automatically push you away from things you think you're supposed to work on, toward things you actually like. "Always produce" will discover your life's work the way water, with the aid of gravity, finds the hole in your roof.

Of course, figuring out what you like to work on doesn't mean you get to work on it. That's a separate question. And if you're ambitious you have to keep them separate: you have to make a conscious effort to keep your ideas about what you want from being contaminated by what seems possible. [6]

It's painful to keep them apart, because it's painful to observe the gap between them. So most people pre-emptively lower their expectations. For example, if you asked random people on the street if they'd like to be able to draw like Leonardo, you'd find most would say something like "Oh, I can't draw." This is more a statement of intention than fact; it means, I'm not going to try. Because the fact is, if you took a random person off the street and somehow got them to work as hard as they possibly could at drawing for the next twenty years, they'd get surprisingly far. But it would require a great moral effort; it would mean staring failure in the eye every day for years. And so to protect themselves people say "I can't."

Another related line you often hear is that not everyone can do work they love — that someone has to do the unpleasant jobs. Really? How do you make them? In the US the only mechanism for forcing people to do unpleasant jobs is the draft, and that hasn't been invoked for over 30 years. All we can do is encourage people to do unpleasant work, with money and prestige.

If there's something people still won't do, it seems as if society just has to make do without. That's what happened with domestic servants. For millennia that was the canonical example of a job "someone had to do." And yet in the mid twentieth century servants practically disappeared in rich countries, and the rich have just had to do without.

So while there may be some things someone has to do, there's a good chance anyone saying that about any particular job is mistaken. Most unpleasant jobs would either get automated or go undone if no one were willing to do them.

Two Routes

There's another sense of "not everyone can do work they love" that's all too true, however. One has to make a living, and it's hard to get paid for doing work you love. There are two routes to that destination:

> The organic route: as you become more eminent, gradually to increase the parts of your job that you like at the expense of those you don't.
>
> The two-job route: to work at things you don't like to get money to work on things you do.

The organic route is more common. It happens naturally to anyone who does good work. A young architect has to take whatever work he can get, but if he does well he'll gradually be in a position to pick and choose among projects. The

disadvantage of this route is that it's slow and uncertain. Even tenure is not real freedom.

The two-job route has several variants depending on how long you work for money at a time. At one extreme is the "day job," where you work regular hours at one job to make money, and work on what you love in your spare time. At the other extreme you work at something till you make enough not to have to work for money again.

The two-job route is less common than the organic route, because it requires a deliberate choice. It's also more dangerous. Life tends to get more expensive as you get older, so it's easy to get sucked into working longer than you expected at the money job. Worse still, anything you work on changes you. If you work too long on tedious stuff, it will rot your brain. And the best paying jobs are most dangerous, because they require your full attention.

The advantage of the two-job route is that it lets you jump over obstacles. The landscape of possible jobs isn't flat; there are walls of varying heights between different kinds of work. [7] The trick of maximizing the parts of your job that you like can get you from architecture to product design, but not, probably, to music. If you make money doing one thing and then work on another, you have more freedom of choice.

Which route should you take? That depends on how sure you are of what you want to do, how good you are at taking orders, how much risk you can stand, and the odds that anyone will pay (in your lifetime) for what you want to do. If you're sure of the general area you want to work in and it's something people are likely to pay you for, then you should probably take the organic route. But if you don't know what you want to work on, or don't like to take orders, you may want to take the two-job route, if you can stand the risk.

Don't decide too soon. Kids who know early what they want to do seem impressive, as if they got the answer to some math question before the other kids. They have an answer, certainly, but odds are it's wrong.

A friend of mine who is a quite successful doctor complains constantly about her job. When people applying to medical school ask her for advice, she wants to shake them and yell "Don't do it!" (But she never does.) How did she get into this fix? In high school she already wanted to be a doctor. And she is so ambitious and determined that she overcame every obstacle along the way — including, unfortunately, not liking it.

Now she has a life chosen for her by a high-school kid.

When you're young, you're given the impression that you'll get enough information to make each choice before you need to make it. But this is certainly not so with work. When you're deciding what to do, you have to operate on ridiculously incomplete information. Even in college you get little idea what various types of

work are like. At best you may have a couple internships, but not all jobs offer internships, and those that do don't teach you much more about the work than being a batboy teaches you about playing baseball.

In the design of lives, as in the design of most other things, you get better results if you use flexible media. So unless you're fairly sure what you want to do, your best bet may be to choose a type of work that could turn into either an organic or two-job career. That was probably part of the reason I chose computers. You can be a professor, or make a lot of money, or morph it into any number of other kinds of work.

It's also wise, early on, to seek jobs that let you do many different things, so you can learn faster what various kinds of work are like. Conversely, the extreme version of the two-job route is dangerous because it teaches you so little about what you like. If you work hard at being a bond trader for ten years, thinking that you'll quit and write novels when you have enough money, what happens when you quit and then discover that you don't actually like writing novels?

Most people would say, I'd take that problem. Give me a million dollars and I'll figure out what to do. But it's harder than it looks. Constraints give your life shape. Remove them and most people have no idea what to do: look at what happens to those who win lotteries or inherit money. Much as everyone thinks they want financial security, the happiest people are not those who have it, but those who like what they do. So a plan that promises freedom at the expense of knowing what to do with it may not be as good as it seems.

Whichever route you take, expect a struggle. Finding work you love is very difficult. Most people fail. Even if you succeed, it's rare to be free to work on what you want till your thirties or forties. But if you have the destination in sight you'll be more likely to arrive at it. If you know you can love work, you're in the home stretch, and if you know what work you love, you're practically there.

Notes

[1] Currently we do the opposite: when we make kids do boring work, like arithmetic drills, instead of admitting frankly that it's boring, we try to disguise it with superficial decorations.

[2] One father told me about a related phenomenon: he found himself concealing from his family how much he liked his work. When he wanted to go to work on a saturday, he found it easier to say that it was because he "had to" for some reason, rather than admitting he preferred to work than stay home with them.

[3] Something similar happens with suburbs. Parents move to suburbs to raise their kids in a safe environment, but suburbs are so dull and artificial that by the time they're fifteen the kids are convinced the whole world is boring.

[4] I'm not saying friends should be the only audience for your work. The more people you can help, the better. But friends should be your compass.

[5] Donald Hall said young would-be poets were mistaken to be so obsessed with being published. But you can imagine what it would do for a 24 year old to get a poem published in *The New Yorker*. Now to people he meets at parties he's a real poet. Actually he's no better or worse than he was before, but to a clueless audience like that, the approval of an official authority makes all the difference. So it's a harder problem than Hall realizes. The reason the young care so much about prestige is that the people they want to impress are not very discerning.

[6] This is isomorphic to the principle that you should prevent your beliefs about how things are from being contaminated by how you wish they were. Most people let them mix pretty promiscuously. The continuing popularity of religion is the most visible index of that.

[7] A more accurate metaphor would be to say that the graph of jobs is not very well connected.

Thanks to Trevor Blackwell, Dan Friedman, Sarah Harlin, Jessica Livingston, Jackie McDonough, Robert Morris, Peter Norvig, David Sloo, and Aaron Swartz for reading drafts of this.

Hebrew Translation

Japanese Translation

Chinese Translation

Russian Translation

Slovak Translation

Italian Translation

German Translation

Spanish Translation

French Translation

Hungarian Translation

Portuguese Translation

Serbian Translation

Greek Translation

Vietnamese Translation

Why YC

March 2006, rev August 2009

Yesterday one of the founders we funded asked me why we started Y Combinator. Or more precisely, he asked if we'd started YC mainly for fun.

Kind of, but not quite. It is enormously fun to be able to work with Rtm and Trevor again. I missed that after we sold Viaweb, and for all the years after I always had a background process running, looking for something we could do together. There is definitely an aspect of a band reunion to Y Combinator. Every couple days I slip and call it "Viaweb."

Viaweb we started very explicitly to make money. I was sick of living from one freelance project to the next, and decided to just work as hard as I could till I'd made enough to solve the problem once and for all. Viaweb was sometimes fun, but it wasn't designed for fun, and mostly it wasn't. I'd be surprised if any startup is. All startups are mostly schleps.

The real reason we started Y Combinator is neither selfish nor virtuous. We didn't start it mainly to make money; we have no idea what our average returns might be, and won't know for years. Nor did we start YC mainly to help out young would-be founders, though we do like the idea, and comfort ourselves occasionally with the thought that if all our investments tank, we will thus have been doing something unselfish. (It's oddly nondeterministic.)

The real reason we started Y Combinator is one probably only a hacker would understand. We did it because it seems such a great hack. There are thousands of smart people who could start companies and don't, and with a relatively small amount of force applied at just the right place, we can spring on the world a stream of new startups that might otherwise not have existed.

In a way this is virtuous, because I think startups are a good thing. But really what motivates us is the completely amoral desire that would motivate any hacker who looked at some complex device and realized that with a tiny tweak he could make it run more efficiently. In this case, the device is the world's economy, which fortunately happens to be open source.

6,631,372

March 2006, rev August 2009

A couple days ago I found to my surprise that I'd been granted a patent. It issued in 2003, but no one told me. I wouldn't know about it now except that a few months ago, while visiting Yahoo, I happened to run into a Big Cheese I knew from working there in the late nineties. He brought up something called Revenue Loop, which Viaweb had been working on when they bought us.

The idea is basically that you sort search results not in order of textual "relevance" (as search engines did then) nor in order of how much advertisers bid (as Overture did) but in order of the bid times the number of transactions. Ordinarily you'd do this for shopping searches, though in fact one of the features of our scheme is that it automatically detects which searches are shopping searches.

If you just order the results in order of bids, you can make the search results useless, because the first results could be dominated by lame sites that had bid the most. But if you order results by bid multiplied by transactions, far from selling out, you're getting a *better* measure of relevance. What could be a better sign that someone was satisfied with a search result than going to the site and buying something?

And, of course, this algorithm automatically maximizes the revenue of the search engine.

Everyone is focused on this type of approach now, but few were in 1998. In 1998 it was all about selling banner ads. We didn't know that, so we were pretty excited when we figured out what seemed to us the optimal way of doing shopping searches.

When Yahoo was thinking of buying us, we had a meeting with Jerry Yang in New York. For him, I now realize, this was supposed to be one of those meetings when you check out a company you've pretty much decided to buy, just to make sure they're ok guys. We weren't expected to do more than chat and seem smart and reasonable. He must have been dismayed when I jumped up to the whiteboard and launched into a presentation of our exciting new technology.

I was just as dismayed when he didn't seem to care at all about it. At the time I thought, "boy, is this guy poker-faced. We present to him what has to be the

optimal way of sorting product search results, and he's not even curious." I didn't realize till much later why he didn't care. In 1998, advertisers were overpaying enormously for ads on web sites. In 1998, if advertisers paid the maximum that traffic was worth to them, Yahoo's revenues would have *decreased.*

Things are different now, of course. Now this sort of thing is all the rage. So when I ran into the Yahoo exec I knew from the old days in the Yahoo cafeteria a few months ago, the first thing he remembered was not (fortunately) all the fights I had with him, but Revenue Loop.

"Well," I said, "I think we actually applied for a patent on it. I'm not sure what happened to the application after I left."

"Really? That would be an important patent."

So someone investigated, and sure enough, that patent application had continued in the pipeline for several years after, and finally issued in 2003.

The main thing that struck me on reading it, actually, is that lawyers at some point messed up my nice clear writing. Some clever person with a spell checker reduced one section to Zen-like incomprehensibility:

> Also, common spelling errors will tend to get fixed. For example, if users searching for "compact disc player" end up spending considerable money at sites offering compact disc players, then those pages will have a higher relevance for that search phrase, even though the phrase "compact disc player" is not present on those pages.

(That "compat disc player" wasn't a typo, guys.)

For the fine prose of the original, see the provisional application of February 1998, back when we were still Viaweb and couldn't afford to pay lawyers to turn every "a lot of" into "considerable."

Are Software Patents Evil?

March 2006

(This essay is derived from a talk at Google.)

A few weeks ago I found to my surprise that I'd been granted four patents. This was all the more surprising because I'd only applied for three. The patents aren't mine, of course. They were assigned to Viaweb, and became Yahoo's when they bought us. But the news set me thinking about the question of software patents generally.

Patents are a hard problem. I've had to advise most of the startups we've funded about them, and despite years of experience I'm still not always sure I'm giving the right advice.

One thing I do feel pretty certain of is that if you're against software patents, you're against patents in general. Gradually our machines consist more and more of software. Things that used to be done with levers and cams and gears are now done with loops and trees and closures. There's nothing special about physical embodiments of control systems that should make them patentable, and the software equivalent not.

Unfortunately, patent law is inconsistent on this point. Patent law in most countries says that algorithms aren't patentable. This rule is left over from a time when "algorithm" meant something like the Sieve of Eratosthenes. In 1800, people could not see as readily as we can that a great many patents on mechanical objects were really patents on the algorithms they embodied.

Patent lawyers still have to pretend that's what they're doing when they patent algorithms. You must not use the word "algorithm" in the title of a patent application, just as you must not use the word "essays" in the title of a book. If you want to patent an algorithm, you have to frame it as a computer system executing that algorithm. Then it's mechanical; phew. The default euphemism for algorithm is "system and method." Try a patent search for that phrase and see how many results you get.

Since software patents are no different from hardware patents, people who say "software patents are evil" are saying simply "patents are evil." So why do so many people complain about software patents specifically?

I think the problem is more with the patent office than the concept of software patents. Whenever software meets government, bad things happen, because software changes fast and government changes slow. The patent office has been overwhelmed by both the volume and the novelty of applications for software patents, and as a result they've made a lot of mistakes.

The most common is to grant patents that shouldn't be granted. To be patentable, an invention has to be more than new. It also has to be non-obvious. And this, especially, is where the USPTO has been dropping the ball. Slashdot has an icon that expresses the problem vividly: a knife and fork with the words "patent pending" superimposed.

The scary thing is, this is the *only* icon they have for patent stories. Slashdot readers now take it for granted that a story about a patent will be about a bogus patent. That's how bad the problem has become.

The problem with Amazon's notorious one-click patent, for example, is not that it's a software patent, but that it's obvious. Any online store that kept people's shipping addresses would have implemented this. The reason Amazon did it first was not that they were especially smart, but because they were one of the earliest sites with enough clout to force customers to log in before they could buy something. [1]

We, as hackers, know the USPTO is letting people patent the knives and forks of our world. The problem is, the USPTO are not hackers. They're probably good at judging new inventions for casting steel or grinding lenses, but they don't understand software yet.

At this point an optimist would be tempted to add "but they will eventually." Unfortunately that might not be true. The problem with software patents is an instance of a more general one: the patent office takes a while to understand new technology. If so, this problem will only get worse, because the rate of technological change seems to be increasing. In thirty years, the patent office may understand the sort of things we now patent as software, but there will be other new types of inventions they understand even less.

Applying for a patent is a negotiation. You generally apply for a broader patent than you think you'll be granted, and the examiners reply by throwing out some of your claims and granting others. So I don't really blame Amazon for applying for the one-click patent. The big mistake was the patent office's, for not insisting on something narrower, with real technical content. By granting such an over-broad patent, the USPTO in effect slept with Amazon on the first date. Was Amazon supposed to say no?

Where Amazon went over to the dark side was not in applying for the patent, but in enforcing it. A lot of companies (Microsoft, for example) have been granted large numbers of preposterously over-broad patents, but they keep them mainly for

defensive purposes. Like nuclear weapons, the main role of big companies' patent portfolios is to threaten anyone who attacks them with a counter-suit. Amazon's suit against Barnes & Noble was thus the equivalent of a nuclear first strike.

That suit probably hurt Amazon more than it helped them. Barnes & Noble was a lame site; Amazon would have crushed them anyway. To attack a rival they could have ignored, Amazon put a lasting black mark on their own reputation. Even now I think if you asked hackers to free-associate about Amazon, the one-click patent would turn up in the first ten topics.

Google clearly doesn't feel that merely holding patents is evil. They've applied for a lot of them. Are they hypocrites? Are patents evil?

There are really two variants of that question, and people answering it often aren't clear in their own minds which they're answering. There's a narrow variant: is it bad, given the current legal system, to apply for patents? and also a broader one: is it bad that the current legal system allows patents?

These are separate questions. For example, in preindustrial societies like medieval Europe, when someone attacked you, you didn't call the police. There were no police. When attacked, you were supposed to fight back, and there were conventions about how to do it. Was this wrong? That's two questions: was it wrong to take justice into your own hands, and was it wrong that you had to? We tend to say yes to the second, but no to the first. If no one else will defend you, you have to defend yourself. [2]

The situation with patents is similar. Business is a kind of ritualized warfare. Indeed, it evolved from actual warfare: most early traders switched on the fly from merchants to pirates depending on how strong you seemed. In business there are certain rules describing how companies may and may not compete with one another, and someone deciding that they're going to play by their own rules is missing the point. Saying "I'm not going to apply for patents just because everyone else does" is not like saying "I'm not going to lie just because everyone else does." It's more like saying "I'm not going to use TCP/IP just because everyone else does." Oh yes you are.

A closer comparison might be someone seeing a hockey game for the first time, realizing with shock that the players were *deliberately* bumping into one another, and deciding that one would on no account be so rude when playing hockey oneself.

Hockey allows checking. It's part of the game. If your team refuses to do it, you simply lose. So it is in business. Under the present rules, patents are part of the game.

What does that mean in practice? We tell the startups we fund not to worry about infringing patents, because startups rarely get sued for patent infringement. There are only two reasons someone might sue you: for money, or to prevent you from

competing with them. Startups are too poor to be worth suing for money. And in practice they don't seem to get sued much by competitors, either. They don't get sued by other startups because (a) patent suits are an expensive distraction, and (b) since the other startups are as young as they are, their patents probably haven't issued yet. [3] Nor do startups, at least in the software business, seem to get sued much by established competitors. Despite all the patents Microsoft holds, I don't know of an instance where they sued a startup for patent infringement. Companies like Microsoft and Oracle don't win by winning lawsuits. That's too uncertain. They win by locking competitors out of their sales channels. If you do manage to threaten them, they're more likely to buy you than sue you.

When you read of big companies filing patent suits against smaller ones, it's usually a big company on the way down, grasping at straws. For example, Unisys's attempts to enforce their patent on LZW compression. When you see a big company threatening patent suits, sell. When a company starts fighting over IP, it's a sign they've lost the real battle, for users.

A company that sues competitors for patent infringement is like a defender who has been beaten so thoroughly that he turns to plead with the referee. You don't do that if you can still reach the ball, even if you genuinely believe you've been fouled. So a company threatening patent suits is a company in trouble.

When we were working on Viaweb, a bigger company in the e-commerce business was granted a patent on online ordering, or something like that. I got a call from a VP there asking if we'd like to license it. I replied that I thought the patent was completely bogus, and would never hold up in court. "Ok," he replied. "So, are you guys hiring?"

If your startup grows big enough, however, you'll start to get sued, no matter what you do. If you go public, for example, you'll be sued by multiple patent trolls who hope you'll pay them off to go away. More on them later.

In other words, no one will sue you for patent infringement till you have money, and once you have money, people will sue you whether they have grounds to or not. So I advise fatalism. Don't waste your time worrying about patent infringement. You're probably violating a patent every time you tie your shoelaces. At the start, at least, just worry about making something great and getting lots of users. If you grow to the point where anyone considers you worth attacking, you're doing well.

We do advise the companies we fund to apply for patents, but not so they can sue competitors. Successful startups either get bought or grow into big companies. If a startup wants to grow into a big company, they should apply for patents to build up the patent portfolio they'll need to maintain an armed truce with other big companies. If they want to get bought, they should apply for patents because patents are part of the mating dance with acquirers.

Most startups that succeed do it by getting bought, and most acquirers care about

patents. Startup acquisitions are usually a build-vs-buy decision for the acquirer. Should we buy this little startup or build our own? And two things, especially, make them decide not to build their own: if you already have a large and rapidly growing user base, and if you have a fairly solid patent application on critical parts of your software.

There's a third reason big companies should prefer buying to building: that if they built their own, they'd screw it up. But few big companies are smart enough yet to admit this to themselves. It's usually the acquirer's engineers who are asked how hard it would be for the company to build their own, and they overestimate their abilities. [4] A patent seems to change the balance. It gives the acquirer an excuse to admit they couldn't copy what you're doing. It may also help them to grasp what's special about your technology.

Frankly, it surprises me how small a role patents play in the software business. It's kind of ironic, considering all the dire things experts say about software patents stifling innovation, but when one looks closely at the software business, the most striking thing is how little patents seem to matter.

In other fields, companies regularly sue competitors for patent infringement. For example, the airport baggage scanning business was for many years a cozy duopoly shared between two companies, InVision and L-3. In 2002 a startup called Reveal appeared, with new technology that let them build scanners a third the size. They were sued for patent infringement before they'd even released a product.

You rarely hear that kind of story in our world. The one example I've found is, embarrassingly enough, Yahoo, which filed a patent suit against a gaming startup called Xfire in 2005. Xfire doesn't seem to be a very big deal, and it's hard to say why Yahoo felt threatened. Xfire's VP of engineering had worked at Yahoo on similar stuff-- in fact, he was listed as an inventor on the patent Yahoo sued over-- so perhaps there was something personal about it. My guess is that someone at Yahoo goofed. At any rate they didn't pursue the suit very vigorously.

Why do patents play so small a role in software? I can think of three possible reasons.

One is that software is so complicated that patents by themselves are not worth very much. I may be maligning other fields here, but it seems that in most types of engineering you can hand the details of some new technique to a group of medium-high quality people and get the desired result. For example, if someone develops a new process for smelting ore that gets a better yield, and you assemble a team of qualified experts and tell them about it, they'll be able to get the same yield. This doesn't seem to work in software. Software is so subtle and unpredictable that "qualified experts" don't get you very far.

That's why we rarely hear phrases like "qualified expert" in the software business. What that level of ability can get you is, say, to make your software compatible

with some other piece of software-- in eight months, at enormous cost. To do anything harder you need individual brilliance. If you assemble a team of qualified experts and tell them to make a new web-based email program, they'll get their asses kicked by a team of inspired nineteen year olds.

Experts can implement, but they can't design. Or rather, expertise in implementation is the only kind most people, including the experts themselves, can measure. [5]

But design is a definite skill. It's not just an airy intangible. Things always seem intangible when you don't understand them. Electricity seemed an airy intangible to most people in 1800. Who knew there was so much to know about it? So it is with design. Some people are good at it and some people are bad at it, and there's something very tangible they're good or bad at.

The reason design counts so much in software is probably that there are fewer constraints than on physical things. Building physical things is expensive and dangerous. The space of possible choices is smaller; you tend to have to work as part of a larger group; and you're subject to a lot of regulations. You don't have any of that if you and a couple friends decide to create a new web-based application.

Because there's so much scope for design in software, a successful application tends to be way more than the sum of its patents. What protects little companies from being copied by bigger competitors is not just their patents, but the thousand little things the big company will get wrong if they try.

The second reason patents don't count for much in our world is that startups rarely attack big companies head-on, the way Reveal did. In the software business, startups beat established companies by transcending them. Startups don't build desktop word processing programs to compete with Microsoft Word. [6] They build Writely. If this paradigm is crowded, just wait for the next one; they run pretty frequently on this route.

Fortunately for startups, big companies are extremely good at denial. If you take the trouble to attack them from an oblique angle, they'll meet you half-way and maneuver to keep you in their blind spot. To sue a startup would mean admitting it was dangerous, and that often means seeing something the big company doesn't want to see. IBM used to sue its mainframe competitors regularly, but they didn't bother much about the microcomputer industry because they didn't want to see the threat it posed. Companies building web based apps are similarly protected from Microsoft, which even now doesn't want to imagine a world in which Windows is irrelevant.

The third reason patents don't seem to matter very much in software is public opinion-- or rather, hacker opinion. In a recent interview, Steve Ballmer coyly left open the possibility of attacking Linux on patent grounds. But I doubt Microsoft would ever be so stupid. They'd face the mother of all boycotts. And not just from

the technical community in general; a lot of their own people would rebel.

Good hackers care a lot about matters of principle, and they are highly mobile. If a company starts misbehaving, smart people won't work there. For some reason this seems to be more true in software than other businesses. I don't think it's because hackers have intrinsically higher principles so much as that their skills are easily transferrable. Perhaps we can split the difference and say that mobility gives hackers the luxury of being principled.

Google's "don't be evil" policy may for this reason be the most valuable thing they've discovered. It's very constraining in some ways. If Google does do something evil, they get doubly whacked for it: once for whatever they did, and again for hypocrisy. But I think it's worth it. It helps them to hire the best people, and it's better, even from a purely selfish point of view, to be constrained by principles than by stupidity.

(I wish someone would get this point across to the present administration.)

I'm not sure what the proportions are of the preceding three ingredients, but the custom among the big companies seems to be not to sue the small ones, and the startups are mostly too busy and too poor to sue one another. So despite the huge number of software patents there's not a lot of suing going on. With one exception: patent trolls.

Patent trolls are companies consisting mainly of lawyers whose whole business is to accumulate patents and threaten to sue companies who actually make things. Patent trolls, it seems safe to say, are evil. I feel a bit stupid saying that, because when you're saying something that Richard Stallman and Bill Gates would both agree with, you must be perilously close to tautologies.

The CEO of Forgent, one of the most notorious patent trolls, says that what his company does is "the American way." Actually that's not true. The American way is to make money by creating wealth, not by suing people. [7] What companies like Forgent do is actually the proto-industrial way. In the period just before the industrial revolution, some of the greatest fortunes in countries like England and France were made by courtiers who extracted some lucrative right from the crown-- like the right to collect taxes on the import of silk-- and then used this to squeeze money from the merchants in that business. So when people compare patent trolls to the mafia, they're more right than they know, because the mafia too are not merely bad, but bad specifically in the sense of being an obsolete business model.

Patent trolls seem to have caught big companies by surprise. In the last couple years they've extracted hundreds of millions of dollars from them. Patent trolls are hard to fight precisely because they create nothing. Big companies are safe from being sued by other big companies because they can threaten a counter-suit. But because patent trolls don't make anything, there's nothing they can be sued for. I predict this loophole will get closed fairly quickly, at least by legal standards. It's clearly an abuse of the system, and the victims are powerful. [8]

But evil as patent trolls are, I don't think they hamper innovation much. They don't sue till a startup has made money, and by that point the innovation that generated it has already happened. I can't think of a startup that avoided working on some problem because of patent trolls.

So much for hockey as the game is played now. What about the more theoretical question of whether hockey would be a better game without checking? Do patents encourage or discourage innovation?

This is a very hard question to answer in the general case. People write whole books on the topic. One of my main hobbies is the history of technology, and even though I've studied the subject for years, it would take me several weeks of research to be able to say whether patents have in general been a net win.

One thing I can say is that 99.9% of the people who express opinions on the subject do it not based on such research, but out of a kind of religious conviction. At least, that's the polite way of putting it; the colloquial version involves speech coming out of organs not designed for that purpose.

Whether they encourage innovation or not, patents were at least intended to. You don't get a patent for nothing. In return for the exclusive right to use an idea, you have to *publish* it, and it was largely to encourage such openness that patents were established.

Before patents, people protected ideas by keeping them secret. With patents, central governments said, in effect, if you tell everyone your idea, we'll protect it for you. There is a parallel here to the rise of civil order, which happened at roughly the same time. Before central governments were powerful enough to enforce order, rich people had private armies. As governments got more powerful, they gradually compelled magnates to cede most responsibility for protecting them. (Magnates still have bodyguards, but no longer to protect them from other magnates.)

Patents, like police, are involved in many abuses. But in both cases the default is something worse. The choice is not "patents or freedom?" any more than it is "police or freedom?" The actual questions are respectively "patents or secrecy?" and "police or gangs?"

As with gangs, we have some idea what secrecy would be like, because that's how things used to be. The economy of medieval Europe was divided up into little tribes, each jealously guarding their privileges and secrets. In Shakespeare's time, "mystery" was synonymous with "craft." Even today we can see an echo of the secrecy of medieval guilds, in the now pointless secrecy of the Masons.

The most memorable example of medieval industrial secrecy is probably Venice, which forbade glassblowers to leave the city, and sent assassins after those who tried. We might like to think we wouldn't go so far, but the movie industry has

already tried to pass laws prescribing three year prison terms just for putting movies on public networks. Want to try a frightening thought experiment? If the movie industry could have any law they wanted, where would they stop? Short of the death penalty, one assumes, but how close would they get?

Even worse than the spectacular abuses might be the overall decrease in efficiency that would accompany increased secrecy. As anyone who has dealt with organizations that operate on a "need to know" basis can attest, dividing information up into little cells is terribly inefficient. The flaw in the "need to know" principle is that you don't *know* who needs to know something. An idea from one area might spark a great discovery in another. But the discoverer doesn't know he needs to know it.

If secrecy were the only protection for ideas, companies wouldn't just have to be secretive with other companies; they'd have to be secretive internally. This would encourage what is already the worst trait of big companies.

I'm not saying secrecy would be worse than patents, just that we couldn't discard patents for free. Businesses would become more secretive to compensate, and in some fields this might get ugly. Nor am I defending the current patent system. There is clearly a lot that's broken about it. But the breakage seems to affect software less than most other fields.

In the software business I know from experience whether patents encourage or discourage innovation, and the answer is the type that people who like to argue about public policy least like to hear: they don't affect innovation much, one way or the other. Most innovation in the software business happens in startups, and startups should simply ignore other companies' patents. At least, that's what we advise, and we bet money on that advice.

The only real role of patents, for most startups, is as an element of the mating dance with acquirers. There patents do help a little. And so they do encourage innovation indirectly, in that they give more power to startups, which is where, pound for pound, the most innovation happens. But even in the mating dance, patents are of secondary importance. It matters more to make something great and get a lot of users.

Notes

[1] You have to be careful here, because a great discovery often seems obvious in retrospect. One-click ordering, however, is not such a discovery.

[2] "Turn the other cheek" skirts the issue; the critical question is not how to deal with slaps, but sword thrusts.

[3] Applying for a patent is now very slow, but it might actually be bad if that got

fixed. At the moment the time it takes to get a patent is conveniently just longer than the time it takes a startup to succeed or fail.

[4] Instead of the canonical "could you build this?" maybe the corp dev guys should be asking "will you build this?" or even "why haven't you already built this?"

[5] Design ability is so hard to measure that you can't even trust the design world's internal standards. You can't assume that someone with a degree in design is any good at design, or that an eminent designer is any better than his peers. If that worked, any company could build products as good as Apple's just by hiring sufficiently qualified designers.

[6] If anyone wanted to try, we'd be interested to hear from them. I suspect it's one of those things that's not as hard as everyone assumes.

[7] Patent trolls can't even claim, like speculators, that they "create" liquidity.

[8] If big companies don't want to wait for the government to take action, there is a way to fight back themselves. For a long time I thought there wasn't, because there was nothing to grab onto. But there is one resource patent trolls need: lawyers. Big technology companies between them generate a lot of legal business. If they agreed among themselves never to do business with any firm employing anyone who had worked for a patent troll, either as an employee or as outside counsel, they could probably starve the trolls of the lawyers they need.

Thanks to Dan Bloomberg, Paul Buchheit, Sarah Harlin, Jessica Livingston, and Peter Norvig for reading drafts of this, to Joel Lehrer and Peter Eng for answering my questions about patents, and to Ankur Pansari for inviting me to speak.

[Japanese Translation]

See Randomness

April 2006, rev August 2009

Plato quotes Socrates as saying "the unexamined life is not worth living." Part of what he meant was that the proper role of humans is to think, just as the proper role of anteaters is to poke their noses into anthills.

A lot of ancient philosophy had the quality — and I don't mean this in an insulting way — of the kind of conversations freshmen have late at night in common rooms:

> What is our purpose? Well, we humans are as conspicuously different from other animals as the anteater. In our case the distinguishing feature is the ability to reason. So obviously that is what we should be doing, and a human who doesn't is doing a bad job of being human — is no better than an animal.

Now we'd give a different answer. At least, someone Socrates's age would. We'd ask why we even suppose we have a "purpose" in life. We may be better adapted for some things than others; we may be happier doing things we're adapted for; but why assume purpose?

The history of ideas is a history of gradually discarding the assumption that it's all about us. No, it turns out, the earth is not the center of the universe — not even the center of the solar system. No, it turns out, humans are not created by God in his own image; they're just one species among many, descended not merely from apes, but from microorganisms. Even the concept of "me" turns out to be fuzzy around the edges if you examine it closely.

The idea that we're the center of things is difficult to discard. So difficult that there's probably room to discard more. Richard Dawkins made another step in that direction only in the last several decades, with the idea of the selfish gene. No, it turns out, we're not even the protagonists: we're just the latest model vehicle our genes have constructed to travel around in. And having kids is our genes heading for the lifeboats. Reading that book snapped my brain out of its previous way of thinking the way Darwin's must have when it first appeared.

(Few people can experience now what Darwin's contemporaries did when *The Origin of Species* was first published, because everyone now is raised either to take evolution for granted, or to regard it as a heresy. No one encounters the idea of natural selection for the first time as an adult.)

So if you want to discover things that have been overlooked till now, one really good place to look is in our blind spot: in our natural, naive belief that it's all about us. And expect to encounter ferocious opposition if you do.

Conversely, if you have to choose between two theories, prefer the one that doesn't center on you.

This principle isn't only for big ideas. It works in everyday life, too. For example, suppose you're saving a piece of cake in the fridge, and you come home one day to find your housemate has eaten it. Two possible theories:

> a) Your housemate did it deliberately to upset you. He *knew* you were saving that piece of cake.
>
> b) Your housemate was hungry.

I say pick b. No one knows who said "never attribute to malice what can be explained by incompetence," but it is a powerful idea. Its more general version is our answer to the Greeks:

> Don't see purpose where there isn't.

Or better still, the positive version:

> See randomness.

- Korean Translation

The Hardest Lessons for Startups to Learn

April 2006

(This essay is derived from a talk at the 2006 Startup School.)

The startups we've funded so far are pretty quick, but they seem quicker to learn some lessons than others. I think it's because some things about startups are kind of counterintuitive.

We've now invested in enough companies that I've learned a trick for determining which points are the counterintuitive ones: they're the ones I have to keep repeating.

So I'm going to number these points, and maybe with future startups I'll be able to pull off a form of Huffman coding. I'll make them all read this, and then instead of nagging them in detail, I'll just be able to say: *number four!*

1. Release Early.

The thing I probably repeat most is this recipe for a startup: get a version 1 out fast, then improve it based on users' reactions.

By "release early" I don't mean you should release something full of bugs, but that you should release something minimal. Users hate bugs, but they don't seem to mind a minimal version 1, if there's more coming soon.

There are several reasons it pays to get version 1 done fast. One is that this is simply the right way to write software, whether for a startup or not. I've been repeating that since 1993, and I haven't seen much since to contradict it. I've seen a lot of startups die because they were too slow to release stuff, and none because they were too quick. [1]

One of the things that will surprise you if you build something popular is that you won't know your users. Reddit now has almost half a million unique visitors a month. Who are all those people? They have no idea. No web startup does. And since you don't know your users, it's dangerous to guess what they'll like. Better to release something and let them tell you.

Wufoo took this to heart and released their form-builder before the underlying database. You can't even drive the thing yet, but 83,000 people came to sit in the driver's seat and hold the steering wheel. And Wufoo got valuable feedback from it: Linux users complained they used too much Flash, so they rewrote their software not to. If they'd waited to release everything at once, they wouldn't have discovered this problem till it was more deeply wired in.

Even if you had no users, it would still be important to release quickly, because for a startup the initial release acts as a shakedown cruise. If anything major is broken-- if the idea's no good, for example, or the founders hate one another-- the stress of getting that first version out will expose it. And if you have such problems you want to find them early.

Perhaps the most important reason to release early, though, is that it makes you work harder. When you're working on something that isn't released, problems are intriguing. In something that's out there, problems are alarming. There is a lot more urgency once you release. And I think that's precisely why people put it off. They know they'll have to work a lot harder once they do. [2]

2. Keep Pumping Out Features.

Of course, "release early" has a second component, without which it would be bad advice. If you're going to start with something that doesn't do much, you better improve it fast.

What I find myself repeating is "pump out features." And this rule isn't just for the initial stages. This is something all startups should do for as long as they want to be considered startups.

I don't mean, of course, that you should make your application ever more complex. By "feature" I mean one unit of hacking-- one quantum of making users' lives better.

As with exercise, improvements beget improvements. If you run every day, you'll probably feel like running tomorrow. But if you skip running for a couple weeks, it will be an effort to drag yourself out. So it is with hacking: the more ideas you implement, the more ideas you'll have. You should make your system better at least in some small way every day or two.

This is not just a good way to get development done; it is also a form of marketing. Users love a site that's constantly improving. In fact, users expect a site to improve. Imagine if you visited a site that seemed very good, and then returned two months later and not one thing had changed. Wouldn't it start to seem lame? [3]

They'll like you even better when you improve in response to their comments, because customers are used to companies ignoring them. If you're the rare

exception-- a company that actually listens-- you'll generate fanatical loyalty. You won't need to advertise, because your users will do it for you.

This seems obvious too, so why do I have to keep repeating it? I think the problem here is that people get used to how things are. Once a product gets past the stage where it has glaring flaws, you start to get used to it, and gradually whatever features it happens to have become its identity. For example, I doubt many people at Yahoo (or Google for that matter) realized how much better web mail could be till Paul Buchheit showed them.

I think the solution is to assume that anything you've made is far short of what it could be. Force yourself, as a sort of intellectual exercise, to keep thinking of improvements. Ok, sure, what you have is perfect. But if you had to change something, what would it be?

If your product seems finished, there are two possible explanations: (a) it is finished, or (b) you lack imagination. Experience suggests (b) is a thousand times more likely.

3. Make Users Happy.

Improving constantly is an instance of a more general rule: make users happy. One thing all startups have in common is that they can't force anyone to do anything. They can't force anyone to use their software, and they can't force anyone to do deals with them. A startup has to sing for its supper. That's why the successful ones make great things. They have to, or die.

When you're running a startup you feel like a little bit of debris blown about by powerful winds. The most powerful wind is users. They can either catch you and loft you up into the sky, as they did with Google, or leave you flat on the pavement, as they do with most startups. Users are a fickle wind, but more powerful than any other. If they take you up, no competitor can keep you down.

As a little piece of debris, the rational thing for you to do is not to lie flat, but to curl yourself into a shape the wind will catch.

I like the wind metaphor because it reminds you how impersonal the stream of traffic is. The vast majority of people who visit your site will be casual visitors. It's them you have to design your site for. The people who really care will find what they want by themselves.

The median visitor will arrive with their finger poised on the Back button. Think about your own experience: most links you follow lead to something lame. Anyone who has used the web for more than a couple weeks has been *trained* to click on Back after following a link. So your site has to say "Wait! Don't click on Back. This site isn't lame. Look at this, for example."

There are two things you have to do to make people pause. The most important is

to explain, as concisely as possible, what the hell your site is about. How often have you visited a site that seemed to assume you already knew what they did? For example, the corporate site that says the company makes

> enterprise content management solutions for business that enable organizations to unify people, content and processes to minimize business risk, accelerate time-to-value and sustain lower total cost of ownership.

An established company may get away with such an opaque description, but no startup can. A startup should be able to explain in one or two sentences exactly what it does. [4] And not just to users. You need this for everyone: investors, acquirers, partners, reporters, potential employees, and even current employees. You probably shouldn't even start a company to do something that can't be described compellingly in one or two sentences.

The other thing I repeat is to give people everything you've got, right away. If you have something impressive, try to put it on the front page, because that's the only one most visitors will see. Though indeed there's a paradox here: the more you push the good stuff toward the front, the more likely visitors are to explore further. [5]

In the best case these two suggestions get combined: you tell visitors what your site is about by *showing* them. One of the standard pieces of advice in fiction writing is "show, don't tell." Don't say that a character's angry; have him grind his teeth, or break his pencil in half. Nothing will explain what your site does so well as using it.

The industry term here is "conversion." The job of your site is to convert casual visitors into users-- whatever your definition of a user is. You can measure this in your growth rate. Either your site is catching on, or it isn't, and you must know which. If you have decent growth, you'll win in the end, no matter how obscure you are now. And if you don't, you need to fix something.

4. Fear the Right Things.

Another thing I find myself saying a lot is "don't worry." Actually, it's more often "don't worry about this; worry about that instead." Startups are right to be paranoid, but they sometimes fear the wrong things.

Most visible disasters are not so alarming as they seem. Disasters are normal in a startup: a founder quits, you discover a patent that covers what you're doing, your servers keep crashing, you run into an insoluble technical problem, you have to change your name, a deal falls through-- these are all par for the course. They won't kill you unless you let them.

Nor will most competitors. A lot of startups worry "what if Google builds something like us?" Actually big companies are not the ones you have to worry about-- not even Google. The people at Google are smart, but no smarter than you; they're

not as motivated, because Google is not going to go out of business if this one product fails; and even at Google they have a lot of bureaucracy to slow them down.

What you should fear, as a startup, is not the established players, but other startups you don't know exist yet. They're way more dangerous than Google because, like you, they're cornered animals.

Looking just at existing competitors can give you a false sense of security. You should compete against what someone else *could* be doing, not just what you can see people doing. A corollary is that you shouldn't relax just because you have no visible competitors yet. No matter what your idea, there's someone else out there working on the same thing.

That's the downside of it being easier to start a startup: more people are doing it. But I disagree with Caterina Fake when she says that makes this a bad time to start a startup. More people are starting startups, but not as many more as could. Most college graduates still think they have to get a job. The average person can't ignore something that's been beaten into their head since they were three just because serving web pages recently got a lot cheaper.

And in any case, competitors are not the biggest threat. Way more startups hose themselves than get crushed by competitors. There are a lot of ways to do it, but the three main ones are internal disputes, inertia, and ignoring users. Each is, by itself, enough to kill you. But if I had to pick the worst, it would be ignoring users. If you want a recipe for a startup that's going to die, here it is: a couple of founders who have some great idea they know everyone is going to love, and that's what they're going to build, no matter what.

Almost everyone's initial plan is broken. If companies stuck to their initial plans, Microsoft would be selling programming languages, and Apple would be selling printed circuit boards. In both cases their customers told them what their business should be-- and they were smart enough to listen.

As Richard Feynman said, the imagination of nature is greater than the imagination of man. You'll find more interesting things by looking at the world than you could ever produce just by thinking. This principle is very powerful. It's why the best abstract painting still falls short of Leonardo, for example. And it applies to startups too. No idea for a product could ever be so clever as the ones you can discover by smashing a beam of prototypes into a beam of users.

5. Commitment Is a Self-Fulfilling Prophecy.

I now have enough experience with startups to be able to say what the most important quality is in a startup founder, and it's not what you might think. The most important quality in a startup founder is determination. Not intelligence--determination.

This is a little depressing. I'd like to believe Viaweb succeeded because we were smart, not merely determined. A lot of people in the startup world want to believe that. Not just founders, but investors too. They like the idea of inhabiting a world ruled by intelligence. And you can tell they really believe this, because it affects their investment decisions.

Time after time VCs invest in startups founded by eminent professors. This may work in biotech, where a lot of startups simply commercialize existing research, but in software you want to invest in students, not professors. Microsoft, Yahoo, and Google were all founded by people who dropped out of school to do it. What students lack in experience they more than make up in dedication.

Of course, if you want to get rich, it's not enough merely to be determined. You have to be smart too, right? I'd like to think so, but I've had an experience that convinced me otherwise: I spent several years living in New York.

You can lose quite a lot in the brains department and it won't kill you. But lose even a little bit in the commitment department, and that will kill you very rapidly.

Running a startup is like walking on your hands: it's possible, but it requires extraordinary effort. If an ordinary employee were asked to do the things a startup founder has to, he'd be very indignant. Imagine if you were hired at some big company, and in addition to writing software ten times faster than you'd ever had to before, they expected you to answer support calls, administer the servers, design the web site, cold-call customers, find the company office space, and go out and get everyone lunch.

And to do all this not in the calm, womb-like atmosphere of a big company, but against a backdrop of constant disasters. That's the part that really demands determination. In a startup, there's always some disaster happening. So if you're the least bit inclined to find an excuse to quit, there's always one right there.

But if you lack commitment, chances are it will have been hurting you long before you actually quit. Everyone who deals with startups knows how important commitment is, so if they sense you're ambivalent, they won't give you much attention. If you lack commitment, you'll just find that for some mysterious reason good things happen to your competitors but not to you. If you lack commitment, it will seem to you that you're unlucky.

Whereas if you're determined to stick around, people will pay attention to you, because odds are they'll have to deal with you later. You're a local, not just a tourist, so everyone has to come to terms with you.

At Y Combinator we sometimes mistakenly fund teams who have the attitude that they're going to give this startup thing a shot for three months, and if something great happens, they'll stick with it-- "something great" meaning either that someone wants to buy them or invest millions of dollars in them. But if this is your attitude, "something great" is very unlikely to happen to you, because both

acquirers and investors judge you by your level of commitment.

If an acquirer thinks you're going to stick around no matter what, they'll be more likely to buy you, because if they don't and you stick around, you'll probably grow, your price will go up, and they'll be left wishing they'd bought you earlier. Ditto for investors. What really motivates investors, even big VCs, is not the hope of good returns, but the fear of missing out. [6] So if you make it clear you're going to succeed no matter what, and the only reason you need them is to make it happen a little faster, you're much more likely to get money.

You can't fake this. The only way to convince everyone that you're ready to fight to the death is actually to be ready to.

You have to be the right kind of determined, though. I carefully chose the word determined rather than stubborn, because stubbornness is a disastrous quality in a startup. You have to be determined, but flexible, like a running back. A successful running back doesn't just put his head down and try to run through people. He improvises: if someone appears in front of him, he runs around them; if someone tries to grab him, he spins out of their grip; he'll even run in the wrong direction briefly if that will help. The one thing he'll never do is stand still. [7]

6. There Is Always Room.

I was talking recently to a startup founder about whether it might be good to add a social component to their software. He said he didn't think so, because the whole social thing was tapped out. Really? So in a hundred years the only social networking sites will be the Facebook, MySpace, Flickr, and Del.icio.us? Not likely.

There is always room for new stuff. At every point in history, even the darkest bits of the dark ages, people were discovering things that made everyone say "why didn't anyone think of that before?" We know this continued to be true up till 2004, when the Facebook was founded-- though strictly speaking someone else did think of that.

The reason we don't see the opportunities all around us is that we adjust to however things are, and assume that's how things have to be. For example, it would seem crazy to most people to try to make a better search engine than Google. Surely that field, at least, is tapped out. Really? In a hundred years-- or even twenty-- are people still going to search for information using something like the current Google? Even Google probably doesn't think that.

In particular, I don't think there's any limit to the number of startups. Sometimes you hear people saying "All these guys starting startups now are going to be disappointed. How many little startups are Google and Yahoo going to buy, after all?" That sounds cleverly skeptical, but I can prove it's mistaken. No one proposes that there's some limit to the number of people who can be employed in an economy consisting of big, slow-moving companies with a couple thousand people each. Why should there be any limit to the number who could be employed by

small, fast-moving companies with ten each? It seems to me the only limit would be the number of people who want to work that hard.

The limit on the number of startups is not the number that can get acquired by Google and Yahoo-- though it seems even that should be unlimited, if the startups were actually worth buying-- but the amount of wealth that can be created. And I don't think there's any limit on that, except cosmological ones.

So for all practical purposes, there is no limit to the number of startups. Startups make wealth, which means they make things people want, and if there's a limit on the number of things people want, we are nowhere near it. I still don't even have a flying car.

7. Don't Get Your Hopes Up.

This is another one I've been repeating since long before Y Combinator. It was practically the corporate motto at Viaweb.

Startup founders are naturally optimistic. They wouldn't do it otherwise. But you should treat your optimism the way you'd treat the core of a nuclear reactor: as a source of power that's also very dangerous. You have to build a shield around it, or it will fry you.

The shielding of a reactor is not uniform; the reactor would be useless if it were. It's pierced in a few places to let pipes in. An optimism shield has to be pierced too. I think the place to draw the line is between what you expect of yourself, and what you expect of other people. It's ok to be optimistic about what you can do, but assume the worst about machines and other people.

This is particularly necessary in a startup, because you tend to be pushing the limits of whatever you're doing. So things don't happen in the smooth, predictable way they do in the rest of the world. Things change suddenly, and usually for the worse.

Shielding your optimism is nowhere more important than with deals. If your startup is doing a deal, just assume it's not going to happen. The VCs who say they're going to invest in you aren't. The company that says they're going to buy you isn't. The big customer who wants to use your system in their whole company won't. Then if things work out you can be pleasantly surprised.

The reason I warn startups not to get their hopes up is not to save them from being *disappointed* when things fall through. It's for a more practical reason: to prevent them from leaning their company against something that's going to fall over, taking them with it.

For example, if someone says they want to invest in you, there's a natural tendency to stop looking for other investors. That's why people proposing deals seem so positive: they *want* you to stop looking. And you want to stop too,

because doing deals is a pain. Raising money, in particular, is a huge time sink. So you have to consciously force yourself to keep looking.

Even if you ultimately do the first deal, it will be to your advantage to have kept looking, because you'll get better terms. Deals are dynamic; unless you're negotiating with someone unusually honest, there's not a single point where you shake hands and the deal's done. There are usually a lot of subsidiary questions to be cleared up after the handshake, and if the other side senses weakness-- if they sense you need this deal-- they will be very tempted to screw you in the details.

VCs and corp dev guys are professional negotiators. They're trained to take advantage of weakness. [8] So while they're often nice guys, they just can't help it. And as pros they do this more than you. So don't even try to bluff them. The only way a startup can have any leverage in a deal is genuinely not to need it. And if you don't believe in a deal, you'll be less likely to depend on it.

So I want to plant a hypnotic suggestion in your heads: when you hear someone say the words "we want to invest in you" or "we want to acquire you," I want the following phrase to appear automatically in your head: *don't get your hopes up.* Just continue running your company as if this deal didn't exist. Nothing is more likely to make it close.

The way to succeed in a startup is to focus on the goal of getting lots of users, and keep walking swiftly toward it while investors and acquirers scurry alongside trying to wave money in your face.

Speed, not Money

The way I've described it, starting a startup sounds pretty stressful. It is. When I talk to the founders of the companies we've funded, they all say the same thing: I knew it would be hard, but I didn't realize it would be this hard.

So why do it? It would be worth enduring a lot of pain and stress to do something grand or heroic, but just to make money? Is making money really that important?

No, not really. It seems ridiculous to me when people take business too seriously. I regard making money as a boring errand to be got out of the way as soon as possible. There is nothing grand or heroic about starting a startup per se.

So why do I spend so much time thinking about startups? I'll tell you why. Economically, a startup is best seen not as a way to get rich, but as a way to work faster. You have to make a living, and a startup is a way to get that done quickly, instead of letting it drag on through your whole life. [9]

We take it for granted most of the time, but human life is fairly miraculous. It is also palpably short. You're given this marvellous thing, and then poof, it's taken away. You can see why people invent gods to explain it. But even to people who don't believe in gods, life commands respect. There are times in most of our lives

when the days go by in a blur, and almost everyone has a sense, when this happens, of wasting something precious. As Ben Franklin said, if you love life, don't waste time, because time is what life is made of.

So no, there's nothing particularly grand about making money. That's not what makes startups worth the trouble. What's important about startups is the speed. By compressing the dull but necessary task of making a living into the smallest possible time, you show respect for life, and there is something grand about that.

Notes

[1] Startups can die from releasing something full of bugs, and not fixing them fast enough, but I don't know of any that died from releasing something stable but minimal very early, then promptly improving it.

[2] I know this is why I haven't released Arc. The moment I do, I'll have people nagging me for features.

[3] A web site is different from a book or movie or desktop application in this respect. Users judge a site not as a single snapshot, but as an animation with multiple frames. Of the two, I'd say the rate of improvement is more important to users than where you currently are.

[4] It should not always tell this to users, however. For example, MySpace is basically a replacement mall for mallrats. But it was wiser for them, initially, to pretend that the site was about bands.

[5] Similarly, don't make users register to try your site. Maybe what you have is so valuable that visitors should gladly register to get at it. But they've been trained to expect the opposite. Most of the things they've tried on the web have sucked-- and probably especially those that made them register.

[6] VCs have rational reasons for behaving this way. They don't make their money (if they make money) off their median investments. In a typical fund, half the companies fail, most of the rest generate mediocre returns, and one or two "make the fund" by succeeding spectacularly. So if they miss just a few of the most promising opportunities, it could hose the whole fund.

[7] The attitude of a running back doesn't translate to soccer. Though it looks great when a forward dribbles past multiple defenders, a player who persists in trying such things will do worse in the long term than one who passes.

[8] The reason Y Combinator never negotiates valuations is that we're not professional negotiators, and don't want to turn into them.

[9] There are two ways to do work you love: (a) to make money, then work on what you love, or (b) to get a job where you get paid to work on stuff you love. In practice the first phases of both consist mostly of unedifying schleps, and in (b) the second phase is less secure.

Thanks to Sam Altman, Trevor Blackwell, Beau Hartshorne, Jessica Livingston, and Robert Morris for reading drafts of this.

▪

Romanian Translation

▪

Russian Translation

▪

French Translation

▪

Japanese Translation

How to Be Silicon Valley

May 2006

(This essay is derived from a keynote at Xtech.)

Could you reproduce Silicon Valley elsewhere, or is there something unique about it?

It wouldn't be surprising if it were hard to reproduce in other countries, because you couldn't reproduce it in most of the US either. What does it take to make a silicon valley even here?

What it takes is the right people. If you could get the right ten thousand people to move from Silicon Valley to Buffalo, Buffalo would become Silicon Valley. [1]

That's a striking departure from the past. Up till a couple decades ago, geography was destiny for cities. All great cities were located on waterways, because cities made money by trade, and water was the only economical way to ship.

Now you could make a great city anywhere, if you could get the right people to move there. So the question of how to make a silicon valley becomes: who are the right people, and how do you get them to move?

Two Types

I think you only need two kinds of people to create a technology hub: rich people and nerds. They're the limiting reagents in the reaction that produces startups, because they're the only ones present when startups get started. Everyone else will move.

Observation bears this out: within the US, towns have become startup hubs if and only if they have both rich people and nerds. Few startups happen in Miami, for example, because although it's full of rich people, it has few nerds. It's not the kind of place nerds like.

Whereas Pittsburgh has the opposite problem: plenty of nerds, but no rich people. The top US Computer Science departments are said to be MIT, Stanford, Berkeley, and Carnegie-Mellon. MIT yielded Route 128. Stanford and Berkeley yielded Silicon Valley. But Carnegie-Mellon? The record skips at that point. Lower down the list,

the University of Washington yielded a high-tech community in Seattle, and the University of Texas at Austin yielded one in Austin. But what happened in Pittsburgh? And in Ithaca, home of Cornell, which is also high on the list?

I grew up in Pittsburgh and went to college at Cornell, so I can answer for both. The weather is terrible, particularly in winter, and there's no interesting old city to make up for it, as there is in Boston. Rich people don't want to live in Pittsburgh or Ithaca. So while there are plenty of hackers who could start startups, there's no one to invest in them.

Not Bureaucrats

Do you really need the rich people? Wouldn't it work to have the government invest in the nerds? No, it would not. Startup investors are a distinct type of rich people. They tend to have a lot of experience themselves in the technology business. This (a) helps them pick the right startups, and (b) means they can supply advice and connections as well as money. And the fact that they have a personal stake in the outcome makes them really pay attention.

Bureaucrats by their nature are the exact opposite sort of people from startup investors. The idea of them making startup investments is comic. It would be like mathematicians running *Vogue*-- or perhaps more accurately, *Vogue* editors running a math journal. [2]

Though indeed, most things bureaucrats do, they do badly. We just don't notice usually, because they only have to compete against other bureaucrats. But as startup investors they'd have to compete against pros with a great deal more experience and motivation.

Even corporations that have in-house VC groups generally forbid them to make their own investment decisions. Most are only allowed to invest in deals where some reputable private VC firm is willing to act as lead investor.

Not Buildings

If you go to see Silicon Valley, what you'll see are buildings. But it's the people that make it Silicon Valley, not the buildings. I read occasionally about attempts to set up "technology parks" in other places, as if the active ingredient of Silicon Valley were the office space. An article about Sophia Antipolis bragged that companies there included Cisco, Compaq, IBM, NCR, and Nortel. Don't the French realize these aren't startups?

Building office buildings for technology companies won't get you a silicon valley, because the key stage in the life of a startup happens before they want that kind of space. The key stage is when they're three guys operating out of an apartment. Wherever the startup is when it gets funded, it will stay. The defining quality of Silicon Valley is not that Intel or Apple or Google have offices there, but that they were *started* there.

So if you want to reproduce Silicon Valley, what you need to reproduce is those two or three founders sitting around a kitchen table deciding to start a company. And to reproduce that you need those people.

Universities

The exciting thing is, *all* you need are the people. If you could attract a critical mass of nerds and investors to live somewhere, you could reproduce Silicon Valley. And both groups are highly mobile. They'll go where life is good. So what makes a place good to them?

What nerds like is other nerds. Smart people will go wherever other smart people are. And in particular, to great universities. In theory there could be other ways to attract them, but so far universities seem to be indispensable. Within the US, there are no technology hubs without first-rate universities-- or at least, first-rate computer science departments.

So if you want to make a silicon valley, you not only need a university, but one of the top handful in the world. It has to be good enough to act as a magnet, drawing the best people from thousands of miles away. And that means it has to stand up to existing magnets like MIT and Stanford.

This sounds hard. Actually it might be easy. My professor friends, when they're deciding where they'd like to work, consider one thing above all: the quality of the other faculty. What attracts professors is good colleagues. So if you managed to recruit, en masse, a significant number of the best young researchers, you could create a first-rate university from nothing overnight. And you could do that for surprisingly little. If you paid 200 people hiring bonuses of $3 million apiece, you could put together a faculty that would bear comparison with any in the world. And from that point the chain reaction would be self-sustaining. So whatever it costs to establish a mediocre university, for an additional half billion or so you could have a great one. [3]

Personality

However, merely creating a new university would not be enough to start a silicon valley. The university is just the seed. It has to be planted in the right soil, or it won't germinate. Plant it in the wrong place, and you just create Carnegie-Mellon.

To spawn startups, your university has to be in a town that has attractions other than the university. It has to be a place where investors want to live, and students want to stay after they graduate.

The two like much the same things, because most startup investors are nerds themselves. So what do nerds look for in a town? Their tastes aren't completely different from other people's, because a lot of the towns they like most in the US are also big tourist destinations: San Francisco, Boston, Seattle. But their tastes

can't be quite mainstream either, because they dislike other big tourist destinations, like New York, Los Angeles, and Las Vegas.

There has been a lot written lately about the "creative class." The thesis seems to be that as wealth derives increasingly from ideas, cities will prosper only if they attract those who have them. That is certainly true; in fact it was the basis of Amsterdam's prosperity 400 years ago.

A lot of nerd tastes they share with the creative class in general. For example, they like well-preserved old neighborhoods instead of cookie-cutter suburbs, and locally-owned shops and restaurants instead of national chains. Like the rest of the creative class, they want to live somewhere with personality.

What exactly is personality? I think it's the feeling that each building is the work of a distinct group of people. A town with personality is one that doesn't feel mass-produced. So if you want to make a startup hub-- or any town to attract the "creative class"-- you probably have to ban large development projects. When a large tract has been developed by a single organization, you can always tell. [4]

Most towns with personality are old, but they don't have to be. Old towns have two advantages: they're denser, because they were laid out before cars, and they're more varied, because they were built one building at a time. You could have both now. Just have building codes that ensure density, and ban large scale developments.

A corollary is that you have to keep out the biggest developer of all: the government. A government that asks "How can we build a silicon valley?" has probably ensured failure by the way they framed the question. You don't build a silicon valley; you let one grow.

Nerds

If you want to attract nerds, you need more than a town with personality. You need a town with the right personality. Nerds are a distinct subset of the creative class, with different tastes from the rest. You can see this most clearly in New York, which attracts a lot of creative people, but few nerds. [5]

What nerds like is the kind of town where people walk around smiling. This excludes LA, where no one walks at all, and also New York, where people walk, but not smiling. When I was in grad school in Boston, a friend came to visit from New York. On the subway back from the airport she asked "Why is everyone smiling?" I looked and they weren't smiling. They just looked like they were compared to the facial expressions she was used to.

If you've lived in New York, you know where these facial expressions come from. It's the kind of place where your mind may be excited, but your body knows it's having a bad time. People don't so much enjoy living there as endure it for the sake of the excitement. And if you like certain kinds of excitement, New York is

incomparable. It's a hub of glamour, a magnet for all the shorter half-life isotopes of style and fame.

Nerds don't care about glamour, so to them the appeal of New York is a mystery. People who like New York will pay a fortune for a small, dark, noisy apartment in order to live in a town where the cool people are really cool. A nerd looks at that deal and sees only: pay a fortune for a small, dark, noisy apartment.

Nerds *will* pay a premium to live in a town where the smart people are really smart, but you don't have to pay as much for that. It's supply and demand: glamour is popular, so you have to pay a lot for it.

Most nerds like quieter pleasures. They like cafes instead of clubs; used bookshops instead of fashionable clothing shops; hiking instead of dancing; sunlight instead of tall buildings. A nerd's idea of paradise is Berkeley or Boulder.

Youth

It's the young nerds who start startups, so it's those specifically the city has to appeal to. The startup hubs in the US are all young-feeling towns. This doesn't mean they have to be new. Cambridge has the oldest town plan in America, but it feels young because it's full of students.

What you can't have, if you want to create a silicon valley, is a large, existing population of stodgy people. It would be a waste of time to try to reverse the fortunes of a declining industrial town like Detroit or Philadelphia by trying to encourage startups. Those places have too much momentum in the wrong direction. You're better off starting with a blank slate in the form of a small town. Or better still, if there's a town young people already flock to, that one.

The Bay Area was a magnet for the young and optimistic for decades before it was associated with technology. It was a place people went in search of something new. And so it became synonymous with California nuttiness. There's still a lot of that there. If you wanted to start a new fad-- a new way to focus one's "energy," for example, or a new category of things not to eat-- the Bay Area would be the place to do it. But a place that tolerates oddness in the search for the new is exactly what you want in a startup hub, because economically that's what startups are. Most good startup ideas seem a little crazy; if they were obviously good ideas, someone would have done them already.

(How many people are going to want computers in their *houses*? What, *another* search engine?)

That's the connection between technology and liberalism. Without exception the high-tech cities in the US are also the most liberal. But it's not because liberals are smarter that this is so. It's because liberal cities tolerate odd ideas, and smart people by definition have odd ideas.

Conversely, a town that gets praised for being "solid" or representing "traditional values" may be a fine place to live, but it's never going to succeed as a startup hub. The 2004 presidential election, though a disaster in other respects, conveniently supplied us with a county-by-county map of such places. [6]

To attract the young, a town must have an intact center. In most American cities the center has been abandoned, and the growth, if any, is in the suburbs. Most American cities have been turned inside out. But none of the startup hubs has: not San Francisco, or Boston, or Seattle. They all have intact centers. [7] My guess is that no city with a dead center could be turned into a startup hub. Young people don't want to live in the suburbs.

Within the US, the two cities I think could most easily be turned into new silicon valleys are Boulder and Portland. Both have the kind of effervescent feel that attracts the young. They're each only a great university short of becoming a silicon valley, if they wanted to.

Time

A great university near an attractive town. Is that all it takes? That was all it took to make the original Silicon Valley. Silicon Valley traces its origins to William Shockley, one of the inventors of the transistor. He did the research that won him the Nobel Prize at Bell Labs, but when he started his own company in 1956 he moved to Palo Alto to do it. At the time that was an odd thing to do. Why did he? Because he had grown up there and remembered how nice it was. Now Palo Alto is suburbia, but then it was a charming college town-- a charming college town with perfect weather and San Francisco only an hour away.

The companies that rule Silicon Valley now are all descended in various ways from Shockley Semiconductor. Shockley was a difficult man, and in 1957 his top people-- "the traitorous eight"-- left to start a new company, Fairchild Semiconductor. Among them were Gordon Moore and Robert Noyce, who went on to found Intel, and Eugene Kleiner, who founded the VC firm Kleiner Perkins. Forty-two years later, Kleiner Perkins funded Google, and the partner responsible for the deal was John Doerr, who came to Silicon Valley in 1974 to work for Intel.

So although a lot of the newest companies in Silicon Valley don't make anything out of silicon, there always seem to be multiple links back to Shockley. There's a lesson here: startups beget startups. People who work for startups start their own. People who get rich from startups fund new ones. I suspect this kind of organic growth is the only way to produce a startup hub, because it's the only way to grow the expertise you need.

That has two important implications. The first is that you need time to grow a silicon valley. The university you could create in a couple years, but the startup community around it has to grow organically. The cycle time is limited by the time it takes a company to succeed, which probably averages about five years.

The other implication of the organic growth hypothesis is that you can't be somewhat of a startup hub. You either have a self-sustaining chain reaction, or not. Observation confirms this too: cities either have a startup scene, or they don't. There is no middle ground. Chicago has the third largest metropolitan area in America. As a source of startups it's negligible compared to Seattle, number 15.

The good news is that the initial seed can be quite small. Shockley Semiconductor, though itself not very successful, was big enough. It brought a critical mass of experts in an important new technology together in a place they liked enough to stay.

Competing

Of course, a would-be silicon valley faces an obstacle the original one didn't: it has to compete with Silicon Valley. Can that be done? Probably.

One of Silicon Valley's biggest advantages is its venture capital firms. This was not a factor in Shockley's day, because VC funds didn't exist. In fact, Shockley Semiconductor and Fairchild Semiconductor were not startups at all in our sense. They were subsidiaries-- of Beckman Instruments and Fairchild Camera and Instrument respectively. Those companies were apparently willing to establish subsidiaries wherever the experts wanted to live.

Venture investors, however, prefer to fund startups within an hour's drive. For one, they're more likely to notice startups nearby. But when they do notice startups in other towns they prefer them to move. They don't want to have to travel to attend board meetings, and in any case the odds of succeeding are higher in a startup hub.

The centralizing effect of venture firms is a double one: they cause startups to form around them, and those draw in more startups through acquisitions. And although the first may be weakening because it's now so cheap to start some startups, the second seems as strong as ever. Three of the most admired "Web 2.0" companies were started outside the usual startup hubs, but two of them have already been reeled in through acquisitions.

Such centralizing forces make it harder for new silicon valleys to get started. But by no means impossible. Ultimately power rests with the founders. A startup with the best people will beat one with funding from famous VCs, and a startup that was sufficiently successful would never have to move. So a town that could exert enough pull over the right people could resist and perhaps even surpass Silicon Valley.

For all its power, Silicon Valley has a great weakness: the paradise Shockley found in 1956 is now one giant parking lot. San Francisco and Berkeley are great, but they're forty miles away. Silicon Valley proper is soul-crushing suburban sprawl. It has fabulous weather, which makes it significantly better than the soul-crushing sprawl of most other American cities. But a competitor that managed to avoid

sprawl would have real leverage. All a city needs is to be the kind of place the next traitorous eight look at and say "I want to stay here," and that would be enough to get the chain reaction started.

Notes

[1] It's interesting to consider how low this number could be made. I suspect five hundred would be enough, even if they could bring no assets with them. Probably just thirty, if I could pick them, would be enough to turn Buffalo into a significant startup hub.

[2] Bureaucrats manage to allocate research funding moderately well, but only because (like an in-house VC fund) they outsource most of the work of selection. A professor at a famous university who is highly regarded by his peers will get funding, pretty much regardless of the proposal. That wouldn't work for startups, whose founders aren't sponsored by organizations, and are often unknowns.

[3] You'd have to do it all at once, or at least a whole department at a time, because people would be more likely to come if they knew their friends were. And you should probably start from scratch, rather than trying to upgrade an existing university, or much energy would be lost in friction.

[4] Hypothesis: Any plan in which multiple independent buildings are gutted or demolished to be "redeveloped" as a single project is a net loss of personality for the city, with the exception of the conversion of buildings not previously public, like warehouses.

[5] A few startups get started in New York, but less than a tenth as many per capita as in Boston, and mostly in less nerdy fields like finance and media.

[6] Some blue counties are false positives (reflecting the remaining power of Democractic party machines), but there are no false negatives. You can safely write off all the red counties.

[7] Some "urban renewal" experts took a shot at destroying Boston's in the 1960s, leaving the area around city hall a bleak wasteland, but most neighborhoods successfully resisted them.

Thanks to Chris Anderson, Trevor Blackwell, Marc Hedlund, Jessica Livingston, Robert Morris, Greg Mcadoo, Fred Wilson, and Stephen Wolfram for reading drafts of this, and to Ed Dumbill for inviting me to speak.

(The second part of this talk became Why Startups Condense in America.)

- [VC Deals by Region]
- [Startup Jobs by Region]
- [They Would Be Gods]
- [Interview: Richard Hodgson]
- [Santa Clara Valley, 1971]
- [Scattered Abroad]
- [Russian Translation]
- [Spanish Translation]

Japanese Translation

Portuguese Translation

Arabic Translation

If you liked this, you may also like ***Hackers & Painters***.

Why Startups Condense in America

May 2006

(This essay is derived from a keynote at Xtech.)

Startups happen in clusters. There are a lot of them in Silicon Valley and Boston, and few in Chicago or Miami. A country that wants startups will probably also have to reproduce whatever makes these clusters form.

I've claimed that the recipe is a great university near a town smart people like. If you set up those conditions within the US, startups will form as inevitably as water droplets condense on a cold piece of metal. But when I consider what it would take to reproduce Silicon Valley in another country, it's clear the US is a particularly humid environment. Startups condense more easily here.

It is by no means a lost cause to try to create a silicon valley in another country. There's room not merely to equal Silicon Valley, but to surpass it. But if you want to do that, you have to understand the advantages startups get from being in America.

1. The US Allows Immigration.

For example, I doubt it would be possible to reproduce Silicon Valley in Japan, because one of Silicon Valley's most distinctive features is immigration. Half the people there speak with accents. And the Japanese don't like immigration. When they think about how to make a Japanese silicon valley, I suspect they unconsciously frame it as how to make one consisting only of Japanese people. This way of framing the question probably guarantees failure.

A silicon valley has to be a mecca for the smart and the ambitious, and you can't have a mecca if you don't let people into it.

Of course, it's not saying much that America is more open to immigration than Japan. Immigration policy is one area where a competitor could do better.

2. The US Is a Rich Country.

I could see India one day producing a rival to Silicon Valley. Obviously they have the right people: you can tell that by the number of Indians in the current Silicon

Valley. The problem with India itself is that it's still so poor.

In poor countries, things we take for granted are missing. A friend of mine visiting India sprained her ankle falling down the steps in a railway station. When she turned to see what had happened, she found the steps were all different heights. In industrialized countries we walk down steps our whole lives and never think about this, because there's an infrastructure that prevents such a staircase from being built.

The US has never been so poor as some countries are now. There have never been swarms of beggars in the streets of American cities. So we have no data about what it takes to get from the swarms-of-beggars stage to the silicon-valley stage. Could you have both at once, or does there have to be some baseline prosperity before you get a silicon valley?

I suspect there is some speed limit to the evolution of an economy. Economies are made out of people, and attitudes can only change a certain amount per generation. [1]

3. The US Is Not (Yet) a Police State.

Another country I could see wanting to have a silicon valley is China. But I doubt they could do it yet either. China still seems to be a police state, and although present rulers seem enlightened compared to the last, even enlightened despotism can probably only get you part way toward being a great economic power.

It can get you factories for building things designed elsewhere. Can it get you the designers, though? Can imagination flourish where people can't criticize the government? Imagination means having odd ideas, and it's hard to have odd ideas about technology without also having odd ideas about politics. And in any case, many technical ideas do have political implications. So if you squash dissent, the back pressure will propagate into technical fields. [2]

Singapore would face a similar problem. Singapore seems very aware of the importance of encouraging startups. But while energetic government intervention may be able to make a port run efficiently, it can't coax startups into existence. A state that bans chewing gum has a long way to go before it could create a San Francisco.

Do you need a San Francisco? Might there not be an alternate route to innovation that goes through obedience and cooperation instead of individualism? Possibly, but I'd bet not. Most imaginative people seem to share a certain prickly independence, whenever and wherever they lived. You see it in Diogenes telling Alexander to get out of his light and two thousand years later in Feynman breaking into safes at Los Alamos. [3] Imaginative people don't want to follow or lead. They're most productive when everyone gets to do what they want.

Ironically, of all rich countries the US has lost the most civil liberties recently. But

I'm not too worried yet. I'm hoping once the present administration is out, the natural openness of American culture will reassert itself.

4. American Universities Are Better.

You need a great university to seed a silicon valley, and so far there are few outside the US. I asked a handful of American computer science professors which universities in Europe were most admired, and they all basically said "Cambridge" followed by a long pause while they tried to think of others. There don't seem to be many universities elsewhere that compare with the best in America, at least in technology.

In some countries this is the result of a deliberate policy. The German and Dutch governments, perhaps from fear of elitism, try to ensure that all universities are roughly equal in quality. The downside is that none are especially good. The best professors are spread out, instead of being concentrated as they are in the US. This probably makes them less productive, because they don't have good colleagues to inspire them. It also means no one university will be good enough to act as a mecca, attracting talent from abroad and causing startups to form around it.

The case of Germany is a strange one. The Germans invented the modern university, and up till the 1930s theirs were the best in the world. Now they have none that stand out. As I was mulling this over, I found myself thinking: "I can understand why German universities declined in the 1930s, after they excluded Jews. But surely they should have bounced back by now." Then I realized: maybe not. There are few Jews left in Germany and most Jews I know would not want to move there. And if you took any great American university and removed the Jews, you'd have some pretty big gaps. So maybe it would be a lost cause trying to create a silicon valley in Germany, because you couldn't establish the level of university you'd need as a seed. [4]

It's natural for US universities to compete with one another because so many are private. To reproduce the quality of American universities you probably also have to reproduce this. If universities are controlled by the central government, log-rolling will pull them all toward the mean: the new Institute of X will end up at the university in the district of a powerful politician, instead of where it should be.

5. You Can Fire People in America.

I think one of the biggest obstacles to creating startups in Europe is the attitude toward employment. The famously rigid labor laws hurt every company, but startups especially, because startups have the least time to spare for bureaucratic hassles.

The difficulty of firing people is a particular problem for startups because they have no redundancy. Every person has to do their job well.

But the problem is more than just that some startup might have a problem firing someone they needed to. Across industries and countries, there's a strong inverse correlation between performance and job security. Actors and directors are fired at the end of each film, so they have to deliver every time. Junior professors are fired by default after a few years unless the university chooses to grant them tenure. Professional athletes know they'll be pulled if they play badly for just a couple games. At the other end of the scale (at least in the US) are auto workers, New York City schoolteachers, and civil servants, who are all nearly impossible to fire. The trend is so clear that you'd have to be willfully blind not to see it.

Performance isn't everything, you say? Well, are auto workers, schoolteachers, and civil servants *happier* than actors, professors, and professional athletes?

European public opinion will apparently tolerate people being fired in industries where they really care about performance. Unfortunately the only industry they care enough about so far is soccer. But that is at least a precedent.

6. In America Work Is Less Identified with Employment.

The problem in more traditional places like Europe and Japan goes deeper than the employment laws. More dangerous is the attitude they reflect: that an employee is a kind of servant, whom the employer has a duty to protect. It used to be that way in America too. In 1970 you were still supposed to get a job with a big company, for whom ideally you'd work your whole career. In return the company would take care of you: they'd try not to fire you, cover your medical expenses, and support you in old age.

Gradually employment has been shedding such paternalistic overtones and becoming simply an economic exchange. But the importance of the new model is not just that it makes it easier for startups to grow. More important, I think, is that it it makes it easier for people to *start* startups.

Even in the US most kids graduating from college still think they're supposed to get jobs, as if you couldn't be productive without being someone's employee. But the less you identify work with employment, the easier it becomes to start a startup. When you see your career as a series of different types of work, instead of a lifetime's service to a single employer, there's less risk in starting your own company, because you're only replacing one segment instead of discarding the whole thing.

The old ideas are so powerful that even the most successful startup founders have had to struggle against them. A year after the founding of Apple, Steve Wozniak still hadn't quit HP. He still planned to work there for life. And when Jobs found someone to give Apple serious venture funding, on the condition that Woz quit, he initially refused, arguing that he'd designed both the Apple I and the Apple II while working at HP, and there was no reason he couldn't continue.

7. America Is Not Too Fussy.

If there are any laws regulating businesses, you can assume larval startups will break most of them, because they don't know what the laws are and don't have time to find out.

For example, many startups in America begin in places where it's not really legal to run a business. Hewlett-Packard, Apple, and Google were all run out of garages. Many more startups, including ours, were initially run out of apartments. If the laws against such things were actually enforced, most startups wouldn't happen.

That could be a problem in fussier countries. If Hewlett and Packard tried running an electronics company out of their garage in Switzerland, the old lady next door would report them to the municipal authorities.

But the worst problem in other countries is probably the effort required just to start a company. A friend of mine started a company in Germany in the early 90s, and was shocked to discover, among many other regulations, that you needed $20,000 in capital to incorporate. That's one reason I'm not typing this on an Apfel laptop. Jobs and Wozniak couldn't have come up with that kind of money in a company financed by selling a VW bus and an HP calculator. We couldn't have started Viaweb either. [5]

Here's a tip for governments that want to encourage startups: read the stories of existing startups, and then try to simulate what would have happened in your country. When you hit something that would have killed Apple, prune it off.

Startups are marginal. They're started by the poor and the timid; they begin in marginal space and spare time; they're started by people who are supposed to be doing something else; and though businesses, their founders often know nothing about business. Young startups are fragile. A society that trims its margins sharply will kill them all.

8. America Has a Large Domestic Market.

What sustains a startup in the beginning is the prospect of getting their initial product out. The successful ones therefore make the first version as simple as possible. In the US they usually begin by making something just for the local market.

This works in America, because the local market is 300 million people. It wouldn't work so well in Sweden. In a small country, a startup has a harder task: they have to sell internationally from the start.

The EU was designed partly to simulate a single, large domestic market. The problem is that the inhabitants still speak many different languages. So a software startup in Sweden is still at a disadvantage relative to one in the US, because they have to deal with internationalization from the beginning. It's significant that the most famous recent startup in Europe, Skype, worked on a problem that was

intrinsically international.

However, for better or worse it looks as if Europe will in a few decades speak a single language. When I was a student in Italy in 1990, few Italians spoke English. Now all educated people seem to be expected to-- and Europeans do not like to seem uneducated. This is presumably a taboo subject, but if present trends continue, French and German will eventually go the way of Irish and Luxembourgish: they'll be spoken in homes and by eccentric nationalists.

9. America Has Venture Funding.

Startups are easier to start in America because funding is easier to get. There are now a few VC firms outside the US, but startup funding doesn't only come from VC firms. A more important source, because it's more personal and comes earlier in the process, is money from individual angel investors. Google might never have got to the point where they could raise millions from VC funds if they hadn't first raised a hundred thousand from Andy Bechtolsheim. And he could help them because he was one of the founders of Sun. This pattern is repeated constantly in startup hubs. It's this pattern that *makes* them startup hubs.

The good news is, all you have to do to get the process rolling is get those first few startups successfully launched. If they stick around after they get rich, startup founders will almost automatically fund and encourage new startups.

The bad news is that the cycle is slow. It probably takes five years, on average, before a startup founder can make angel investments. And while governments *might* be able to set up local VC funds by supplying the money themselves and recruiting people from existing firms to run them, only organic growth can produce angel investors.

Incidentally, America's private universities are one reason there's so much venture capital. A lot of the money in VC funds comes from their endowments. So another advantage of private universities is that a good chunk of the country's wealth is managed by enlightened investors.

10. America Has Dynamic Typing for Careers.

Compared to other industrialized countries the US is disorganized about routing people into careers. For example, in America people often don't decide to go to medical school till they've finished college. In Europe they generally decide in high school.

The European approach reflects the old idea that each person has a single, definite occupation-- which is not far from the idea that each person has a natural "station" in life. If this were true, the most efficient plan would be to discover each person's station as early as possible, so they could receive the training appropriate to it.

In the US things are more haphazard. But that turns out to be an advantage as an

economy gets more liquid, just as dynamic typing turns out to work better than static for ill-defined problems. This is particularly true with startups. "Startup founder" is not the sort of career a high school student would choose. If you ask at that age, people will choose conservatively. They'll choose well-understood occupations like engineer, or doctor, or lawyer.

Startups are the kind of thing people don't plan, so you're more likely to get them in a society where it's ok to make career decisions on the fly.

For example, in theory the purpose of a PhD program is to train you to do research. But fortunately in the US this is another rule that isn't very strictly enforced. In the US most people in CS PhD programs are there simply because they wanted to learn more. They haven't decided what they'll do afterward. So American grad schools spawn a lot of startups, because students don't feel they're failing if they don't go into research.

Those worried about America's "competitiveness" often suggest spending more on public schools. But perhaps America's lousy public schools have a hidden advantage. Because they're so bad, the kids adopt an attitude of waiting for college. I did; I knew I was learning so little that I wasn't even learning what the choices were, let alone which to choose. This is demoralizing, but it does at least make you keep an open mind.

Certainly if I had to choose between bad high schools and good universities, like the US, and good high schools and bad universities, like most other industrialized countries, I'd take the US system. Better to make everyone feel like a late bloomer than a failed child prodigy.

Attitudes

There's one item conspicuously missing from this list: American attitudes. Americans are said to be more entrepreneurial, and less afraid of risk. But America has no monopoly on this. Indians and Chinese seem plenty entrepreneurial, perhaps more than Americans.

Some say Europeans are less energetic, but I don't believe it. I think the problem with Europe is not that they lack balls, but that they lack examples.

Even in the US, the most successful startup founders are often technical people who are quite timid, initially, about the idea of starting their own company. Few are the sort of backslapping extroverts one thinks of as typically American. They can usually only summon up the activation energy to start a startup when they meet people who've done it and realize they could too.

I think what holds back European hackers is simply that they don't meet so many people who've done it. You see that variation even within the US. Stanford students are more entrepreneurial than Yale students, but not because of some difference in their characters; the Yale students just have fewer examples.

I admit there seem to be different attitudes toward ambition in Europe and the US. In the US it's ok to be overtly ambitious, and in most of Europe it's not. But this can't be an intrinsically European quality; previous generations of Europeans were as ambitious as Americans. What happened? My hypothesis is that ambition was discredited by the terrible things ambitious people did in the first half of the twentieth century. Now swagger is out. (Even now the image of a very ambitious German presses a button or two, doesn't it?)

It would be surprising if European attitudes weren't affected by the disasters of the twentieth century. It takes a while to be optimistic after events like that. But ambition is human nature. Gradually it will re-emerge. [6]

How To Do Better

I don't mean to suggest by this list that America is the perfect place for startups. It's the best place so far, but the sample size is small, and "so far" is not very long. On historical time scales, what we have now is just a prototype.

So let's look at Silicon Valley the way you'd look at a product made by a competitor. What weaknesses could you exploit? How could you make something users would like better? The users in this case are those critical few thousand people you'd like to move to your silicon valley.

To start with, Silicon Valley is too far from San Francisco. Palo Alto, the original ground zero, is about thirty miles away, and the present center more like forty. So people who come to work in Silicon Valley face an unpleasant choice: either live in the boring sprawl of the valley proper, or live in San Francisco and endure an hour commute each way.

The best thing would be if the silicon valley were not merely closer to the interesting city, but interesting itself. And there is a lot of room for improvement here. Palo Alto is not so bad, but everything built since is the worst sort of strip development. You can measure how demoralizing it is by the number of people who will sacrifice two hours a day commuting rather than live there.

Another area in which you could easily surpass Silicon Valley is public transportation. There is a train running the length of it, and by American standards it's not bad. Which is to say that to Japanese or Europeans it would seem like something out of the third world.

The kind of people you want to attract to your silicon valley like to get around by train, bicycle, and on foot. So if you want to beat America, design a town that puts cars last. It will be a while before any American city can bring itself to do that.

Capital Gains

There are also a couple things you could do to beat America at the national level.

One would be to have lower capital gains taxes. It doesn't seem critical to have the lowest *income* taxes, because to take advantage of those, people have to move. [7] But if capital gains rates vary, you move assets, not yourself, so changes are reflected at market speeds. The lower the rate, the cheaper it is to buy stock in growing companies as opposed to real estate, or bonds, or stocks bought for the dividends they pay.

So if you want to encourage startups you should have a low rate on capital gains. Politicians are caught between a rock and a hard place here, however: make the capital gains rate low and be accused of creating "tax breaks for the rich," or make it high and starve growing companies of investment capital. As Galbraith said, politics is a matter of choosing between the unpalatable and the disastrous. A lot of governments experimented with the disastrous in the twentieth century; now the trend seems to be toward the merely unpalatable.

Oddly enough, the leaders now are European countries like Belgium, which has a capital gains tax rate of zero.

Immigration

The other place you could beat the US would be with smarter immigration policy. There are huge gains to be made here. Silicon valleys are made of people, remember.

Like a company whose software runs on Windows, those in the current Silicon Valley are all too aware of the shortcomings of the INS, but there's little they can do about it. They're hostages of the platform.

America's immigration system has never been well run, and since 2001 there has been an additional admixture of paranoia. What fraction of the smart people who want to come to America can even get in? I doubt even half. Which means if you made a competing technology hub that let in all smart people, you'd immediately get more than half the world's top talent, for free.

US immigration policy is particularly ill-suited to startups, because it reflects a model of work from the 1970s. It assumes good technical people have college degrees, and that work means working for a big company.

If you don't have a college degree you can't get an H1B visa, the type usually issued to programmers. But a test that excludes Steve Jobs, Bill Gates, and Michael Dell can't be a good one. Plus you can't get a visa for working on your own company, only for working as an employee of someone else's. And if you want to apply for citizenship you daren't work for a startup at all, because if your sponsor goes out of business, you have to start over.

American immigration policy keeps out most smart people, and channels the rest into unproductive jobs. It would be easy to do better. Imagine if, instead, you treated immigration like recruiting-- if you made a conscious effort to seek out the

smartest people and get them to come to your country.

A country that got immigration right would have a huge advantage. At this point you could become a mecca for smart people simply by having an immigration system that let them in.

A Good Vector

If you look at the kinds of things you have to do to create an environment where startups condense, none are great sacrifices. Great universities? Livable towns? Civil liberties? Flexible employment laws? Immigration policies that let in smart people? Tax laws that encourage growth? It's not as if you have to risk destroying your country to get a silicon valley; these are all good things in their own right.

And then of course there's the question, can you afford not to? I can imagine a future in which the default choice of ambitious young people is to start their own company rather than work for someone else's. I'm not sure that will happen, but it's where the trend points now. And if that is the future, places that don't have startups will be a whole step behind, like those that missed the Industrial Revolution.

Notes

[1] On the verge of the Industrial Revolution, England was already the richest country in the world. As far as such things can be compared, per capita income in England in 1750 was higher than India's in 1960.

Deane, Phyllis, *The First Industrial Revolution*, Cambridge University Press, 1965.

[2] This has already happened once in China, during the Ming Dynasty, when the country turned its back on industrialization at the command of the court. One of Europe's advantages was that it had no government powerful enough to do that.

[3] Of course, Feynman and Diogenes were from adjacent traditions, but Confucius, though more polite, was no more willing to be told what to think.

[4] For similar reasons it might be a lost cause to try to establish a silicon valley in Israel. Instead of no Jews moving there, only Jews would move there, and I don't think you could build a silicon valley out of just Jews any more than you could out of just Japanese.

(This is not a remark about the qualities of these groups, just their sizes. Japanese are only about 2% of the world population, and Jews about .2%.)

[5] According to the World Bank, the initial capital requirement for German companies is 47.6% of the per capita income. Doh.

World Bank, *Doing Business in 2006*, http://doingbusiness.org

[6] For most of the twentieth century, Europeans looked back on the summer of 1914 as if they'd been living in a dream world. It seems more accurate (or at least, as accurate) to call the years after 1914 a nightmare than to call those before a dream. A lot of the optimism Europeans consider distinctly American is simply what they too were feeling in 1914.

[7] The point where things start to go wrong seems to be about 50%. Above that people get serious about tax avoidance. The reason is that the payoff for avoiding tax grows hyperexponentially ($x/1-x$ for $0 < x < 1$). If your income tax rate is 10%, moving to Monaco would only give you 11% more income, which wouldn't even cover the extra cost. If it's 90%, you'd get ten times as much income. And at 98%, as it was briefly in Britain in the 70s, moving to Monaco would give you fifty times as much income. It seems quite likely that European governments of the 70s never drew this curve.

Thanks to Trevor Blackwell, Matthias Felleisen, Jessica Livingston, Robert Morris, Neil Rimer, Hugues Steinier, Brad Templeton, Fred Wilson, and Stephen Wolfram for reading drafts of this, and to Ed Dumbill for inviting me to speak.

French Translation

Russian Translation

Japanese Translation

Arabic Translation

The Power of the Marginal

June 2006

(This essay is derived from talks at Usenix 2006 and Railsconf 2006.)

A couple years ago my friend Trevor and I went to look at the Apple garage. As we stood there, he said that as a kid growing up in Saskatchewan he'd been amazed at the dedication Jobs and Wozniak must have had to work in a garage.

"Those guys must have been freezing!"

That's one of California's hidden advantages: the mild climate means there's lots of marginal space. In cold places that margin gets trimmed off. There's a sharper line between outside and inside, and only projects that are officially sanctioned — by organizations, or parents, or wives, or at least by oneself — get proper indoor space. That raises the activation energy for new ideas. You can't just tinker. You have to justify.

Some of Silicon Valley's most famous companies began in garages: Hewlett-Packard in 1938, Apple in 1976, Google in 1998. In Apple's case the garage story is a bit of an urban legend. Woz says all they did there was assemble some computers, and that he did all the actual design of the Apple I and Apple II in his apartment or his cube at HP. [1] This was apparently too marginal even for Apple's PR people.

By conventional standards, Jobs and Wozniak were marginal people too. Obviously they were smart, but they can't have looked good on paper. They were at the time a pair of college dropouts with about three years of school between them, and hippies to boot. Their previous business experience consisted of making "blue boxes" to hack into the phone system, a business with the rare distinction of being both illegal and unprofitable.

Outsiders

Now a startup operating out of a garage in Silicon Valley would feel part of an exalted tradition, like the poet in his garret, or the painter who can't afford to heat his studio and thus has to wear a beret indoors. But in 1976 it didn't seem so cool. The world hadn't yet realized that starting a computer company was in the same

category as being a writer or a painter. It hadn't been for long. Only in the preceding couple years had the dramatic fall in the cost of hardware allowed outsiders to compete.

In 1976, everyone looked down on a company operating out of a garage, including the founders. One of the first things Jobs did when they got some money was to rent office space. He wanted Apple to seem like a real company.

They already had something few real companies ever have: a fabulously well designed product. You'd think they'd have had more confidence. But I've talked to a lot of startup founders, and it's always this way. They've built something that's going to change the world, and they're worried about some nit like not having proper business cards.

That's the paradox I want to explore: great new things often come from the margins, and yet the people who discover them are looked down on by everyone, including themselves.

It's an old idea that new things come from the margins. I want to examine its internal structure. Why do great ideas come from the margins? What kind of ideas? And is there anything we can do to encourage the process?

Insiders

One reason so many good ideas come from the margin is simply that there's so much of it. There have to be more outsiders than insiders, if insider means anything. If the number of outsiders is huge it will always seem as if a lot of ideas come from them, even if few do per capita. But I think there's more going on than this. There are real disadvantages to being an insider, and in some kinds of work they can outweigh the advantages.

Imagine, for example, what would happen if the government decided to commission someone to write an official Great American Novel. First there'd be a huge ideological squabble over who to choose. Most of the best writers would be excluded for having offended one side or the other. Of the remainder, the smart ones would refuse such a job, leaving only a few with the wrong sort of ambition. The committee would choose one at the height of his career — that is, someone whose best work was behind him — and hand over the project with copious free advice about how the book should show in positive terms the strength and diversity of the American people, etc, etc.

The unfortunate writer would then sit down to work with a huge weight of expectation on his shoulders. Not wanting to blow such a public commission, he'd play it safe. This book had better command respect, and the way to ensure that would be to make it a tragedy. Audiences have to be enticed to laugh, but if you kill people they feel obliged to take you seriously. As everyone knows, America plus tragedy equals the Civil War, so that's what it would have to be about. When finally completed twelve years later, the book would be a 900-page pastiche of existing

popular novels — roughly *Gone with the Wind* plus *Roots*. But its bulk and celebrity would make it a bestseller for a few months, until blown out of the water by a talk-show host's autobiography. The book would be made into a movie and thereupon forgotten, except by the more waspish sort of reviewers, among whom it would be a byword for bogusness like Milli Vanilli or *Battlefield Earth*.

Maybe I got a little carried away with this example. And yet is this not at each point the way such a project would play out? The government knows better than to get into the novel business, but in other fields where they have a natural monopoly, like nuclear waste dumps, aircraft carriers, and regime change, you'd find plenty of projects isomorphic to this one — and indeed, plenty that were less successful.

This little thought experiment suggests a few of the disadvantages of insider projects: the selection of the wrong kind of people, the excessive scope, the inability to take risks, the need to seem serious, the weight of expectations, the power of vested interests, the undiscerning audience, and perhaps most dangerous, the tendency of such work to become a duty rather than a pleasure.

Tests

A world with outsiders and insiders implies some kind of test for distinguishing between them. And the trouble with most tests for selecting elites is that there are two ways to pass them: to be good at what they try to measure, and to be good at hacking the test itself.

So the first question to ask about a field is how honest its tests are, because this tells you what it means to be an outsider. This tells you how much to trust your instincts when you disagree with authorities, whether it's worth going through the usual channels to become one yourself, and perhaps whether you want to work in this field at all.

Tests are least hackable when there are consistent standards for quality, and the people running the test really care about its integrity. Admissions to PhD programs in the hard sciences are fairly honest, for example. The professors will get whoever they admit as their own grad students, so they try hard to choose well, and they have a fair amount of data to go on. Whereas undergraduate admissions seem to be much more hackable.

One way to tell whether a field has consistent standards is the overlap between the leading practitioners and the people who teach the subject in universities. At one end of the scale you have fields like math and physics, where nearly all the teachers are among the best practitioners. In the middle are medicine, law, history, architecture, and computer science, where many are. At the bottom are business, literature, and the visual arts, where there's almost no overlap between the teachers and the leading practitioners. It's this end that gives rise to phrases like "those who can't do, teach."

Incidentally, this scale might be helpful in deciding what to study in college. When I was in college the rule seemed to be that you should study whatever you were most interested in. But in retrospect you're probably better off studying something moderately interesting with someone who's good at it than something very interesting with someone who isn't. You often hear people say that you shouldn't major in business in college, but this is actually an instance of a more general rule: don't learn things from teachers who are bad at them.

How much you should worry about being an outsider depends on the quality of the insiders. If you're an amateur mathematician and think you've solved a famous open problem, better go back and check. When I was in grad school, a friend in the math department had the job of replying to people who sent in proofs of Fermat's last theorem and so on, and it did not seem as if he saw it as a valuable source of tips — more like manning a mental health hotline. Whereas if the stuff you're writing seems different from what English professors are interested in, that's not necessarily a problem.

Anti-Tests

Where the method of selecting the elite is thoroughly corrupt, most of the good people will be outsiders. In art, for example, the image of the poor, misunderstood genius is not just one possible image of a great artist: it's the *standard* image. I'm not saying it's correct, incidentally, but it is telling how well this image has stuck. You couldn't make a rap like that stick to math or medicine. [2]

If it's corrupt enough, a test becomes an anti-test, filtering out the people it should select by making them to do things only the wrong people would do. Popularity in high school seems to be such a test. There are plenty of similar ones in the grownup world. For example, rising up through the hierarchy of the average big company demands an attention to politics few thoughtful people could spare. [3] Someone like Bill Gates can grow a company under him, but it's hard to imagine him having the patience to climb the corporate ladder at General Electric — or Microsoft, actually.

It's kind of strange when you think about it, because lord-of-the-flies schools and bureaucratic companies are both the default. There are probably a lot of people who go from one to the other and never realize the whole world doesn't work this way.

I think that's one reason big companies are so often blindsided by startups. People at big companies don't realize the extent to which they live in an environment that is one large, ongoing test for the wrong qualities.

If you're an outsider, your best chances for beating insiders are obviously in fields where corrupt tests select a lame elite. But there's a catch: if the tests are corrupt, your victory won't be recognized, at least in your lifetime. You may feel you don't need that, but history suggests it's dangerous to work in fields with corrupt tests. You may beat the insiders, and yet not do as good work, on an absolute scale, as

you would in a field that was more honest.

Standards in art, for example, were almost as corrupt in the first half of the eighteenth century as they are today. This was the era of those fluffy idealized portraits of countesses with their lapdogs. Chardin decided to skip all that and paint ordinary things as he saw them. He's now considered the best of that period — and yet not the equal of Leonardo or Bellini or Memling, who all had the additional encouragement of honest standards.

It can be worth participating in a corrupt contest, however, if it's followed by another that isn't corrupt. For example, it would be worth competing with a company that can spend more than you on marketing, as long as you can survive to the next round, when customers compare your actual products. Similarly, you shouldn't be discouraged by the comparatively corrupt test of college admissions, because it's followed immediately by less hackable tests. [4]

Risk

Even in a field with honest tests, there are still advantages to being an outsider. The most obvious is that outsiders have nothing to lose. They can do risky things, and if they fail, so what? Few will even notice.

The eminent, on the other hand, are weighed down by their eminence. Eminence is like a suit: it impresses the wrong people, and it constrains the wearer.

Outsiders should realize the advantage they have here. Being able to take risks is hugely valuable. Everyone values safety too much, both the obscure and the eminent. No one wants to look like a fool. But it's very useful to be able to. If most of your ideas aren't stupid, you're probably being too conservative. You're not bracketing the problem.

Lord Acton said we should judge talent at its best and character at its worst. For example, if you write one great book and ten bad ones, you still count as a great writer — or at least, a better writer than someone who wrote eleven that were merely good. Whereas if you're a quiet, law-abiding citizen most of the time but occasionally cut someone up and bury them in your backyard, you're a bad guy.

Almost everyone makes the mistake of treating ideas as if they were indications of character rather than talent — as if having a stupid idea made you stupid. There's a huge weight of tradition advising us to play it safe. "Even a fool is thought wise if he keeps silent," says the Old Testament (Proverbs 17:28).

Well, that may be fine advice for a bunch of goatherds in Bronze Age Palestine. There conservatism would be the order of the day. But times have changed. It might still be reasonable to stick with the Old Testament in political questions, but materially the world now has a lot more state. Tradition is less of a guide, not just because things change faster, but because the space of possibilities is so large. The more complicated the world gets, the more valuable it is to be willing to look like a

fool.

Delegation

And yet the more successful people become, the more heat they get if they screw up — or even seem to screw up. In this respect, as in many others, the eminent are prisoners of their own success. So the best way to understand the advantages of being an outsider may be to look at the disadvantages of being an insider.

If you ask eminent people what's wrong with their lives, the first thing they'll complain about is the lack of time. A friend of mine at Google is fairly high up in the company and went to work for them long before they went public. In other words, he's now rich enough not to have to work. I asked him if he could still endure the annoyances of having a job, now that he didn't have to. And he said that there weren't really any annoyances, except — and he got a wistful look when he said this — that he got *so much email*.

The eminent feel like everyone wants to take a bite out of them. The problem is so widespread that people pretending to be eminent do it by pretending to be overstretched.

The lives of the eminent become scheduled, and that's not good for thinking. One of the great advantages of being an outsider is long, uninterrupted blocks of time. That's what I remember about grad school: apparently endless supplies of time, which I spent worrying about, but not writing, my dissertation. Obscurity is like health food — unpleasant, perhaps, but good for you. Whereas fame tends to be like the alcohol produced by fermentation. When it reaches a certain concentration, it kills off the yeast that produced it.

The eminent generally respond to the shortage of time by turning into managers. They don't have time to work. They're surrounded by junior people they're supposed to help or supervise. The obvious solution is to have the junior people do the work. Some good stuff happens this way, but there are problems it doesn't work so well for: the kind where it helps to have everything in one head.

For example, it recently emerged that the famous glass artist Dale Chihuly hasn't actually blown glass for 27 years. He has assistants do the work for him. But one of the most valuable sources of ideas in the visual arts is the resistance of the medium. That's why oil paintings look so different from watercolors. In principle you could make any mark in any medium; in practice the medium steers you. And if you're no longer doing the work yourself, you stop learning from this.

So if you want to beat those eminent enough to delegate, one way to do it is to take advantage of direct contact with the medium. In the arts it's obvious how: blow your own glass, edit your own films, stage your own plays. And in the process pay close attention to accidents and to new ideas you have on the fly. This technique can be generalized to any sort of work: if you're an outsider, don't be ruled by plans. Planning is often just a weakness forced on those who delegate.

Is there a general rule for finding problems best solved in one head? Well, you can manufacture them by taking any project usually done by multiple people and trying to do it all yourself. Wozniak's work was a classic example: he did everything himself, hardware and software, and the result was miraculous. He claims not one bug was ever found in the Apple II, in either hardware or software.

Another way to find good problems to solve in one head is to focus on the grooves in the chocolate bar — the places where tasks are divided when they're split between several people. If you want to beat delegation, focus on a vertical slice: for example, be both writer and editor, or both design buildings and construct them.

One especially good groove to span is the one between tools and things made with them. For example, programming languages and applications are usually written by different people, and this is responsible for a lot of the worst flaws in programming languages. I think every language should be designed simultaneously with a large application written in it, the way C was with Unix.

Techniques for competing with delegation translate well into business, because delegation is endemic there. Instead of avoiding it as a drawback of senility, many companies embrace it as a sign of maturity. In big companies software is often designed, implemented, and sold by three separate types of people. In startups one person may have to do all three. And though this feels stressful, it's one reason startups win. The needs of customers and the means of satisfying them are all in one head.

Focus

The very skill of insiders can be a weakness. Once someone is good at something, they tend to spend all their time doing that. This kind of focus is very valuable, actually. Much of the skill of experts is the ability to ignore false trails. But focus has drawbacks: you don't learn from other fields, and when a new approach arrives, you may be the last to notice.

For outsiders this translates into two ways to win. One is to work on a variety of things. Since you can't derive as much benefit (yet) from a narrow focus, you may as well cast a wider net and derive what benefit you can from similarities between fields. Just as you can compete with delegation by working on larger vertical slices, you can compete with specialization by working on larger horizontal slices — by both writing and illustrating your book, for example.

The second way to compete with focus is to see what focus overlooks. In particular, new things. So if you're not good at anything yet, consider working on something so new that no one else is either. It won't have any prestige yet, if no one is good at it, but you'll have it all to yourself.

The potential of a new medium is usually underestimated, precisely because no

one has yet explored its possibilities. Before Durer tried making engravings, no one took them very seriously. Engraving was for making little devotional images — basically fifteenth century baseball cards of saints. Trying to make masterpieces in this medium must have seemed to Durer's contemporaries the way that, say, making masterpieces in comics might seem to the average person today.

In the computer world we get not new mediums but new platforms: the minicomputer, the microprocessor, the web-based application. At first they're always dismissed as being unsuitable for real work. And yet someone always decides to try anyway, and it turns out you can do more than anyone expected. So in the future when you hear people say of a new platform: yeah, it's popular and cheap, but not ready yet for real work, jump on it.

As well as being more comfortable working on established lines, insiders generally have a vested interest in perpetuating them. The professor who made his reputation by discovering some new idea is not likely to be the one to discover its replacement. This is particularly true with companies, who have not only skill and pride anchoring them to the status quo, but money as well. The Achilles heel of successful companies is their inability to cannibalize themselves. Many innovations consist of replacing something with a cheaper alternative, and companies just don't want to see a path whose immediate effect is to cut an existing source of revenue.

So if you're an outsider you should actively seek out contrarian projects. Instead of working on things the eminent have made prestigious, work on things that could steal that prestige.

The really juicy new approaches are not the ones insiders reject as impossible, but those they ignore as undignified. For example, after Wozniak designed the Apple II he offered it first to his employer, HP. They passed. One of the reasons was that, to save money, he'd designed the Apple II to use a TV as a monitor, and HP felt they couldn't produce anything so declasse.

Less

Wozniak used a TV as a monitor for the simple reason that he couldn't afford a monitor. Outsiders are not merely free but compelled to make things that are cheap and lightweight. And both are good bets for growth: cheap things spread faster, and lightweight things evolve faster.

The eminent, on the other hand, are almost forced to work on a large scale. Instead of garden sheds they must design huge art museums. One reason they work on big things is that they can: like our hypothetical novelist, they're flattered by such opportunities. They also know that big projects will by their sheer bulk impress the audience. A garden shed, however lovely, would be easy to ignore; a few might even snicker at it. You can't snicker at a giant museum, no matter how much you dislike it. And finally, there are all those people the eminent have working for them; they have to choose projects that can keep them all busy.

Outsiders are free of all this. They can work on small things, and there's something very pleasing about small things. Small things can be perfect; big ones always have something wrong with them. But there's a magic in small things that goes beyond such rational explanations. All kids know it. Small things have more personality.

Plus making them is more fun. You can do what you want; you don't have to satisfy committees. And perhaps most important, small things can be done fast. The prospect of seeing the finished project hangs in the air like the smell of dinner cooking. If you work fast, maybe you could have it done tonight.

Working on small things is also a good way to learn. The most important kinds of learning happen one project at a time. ("Next time, I won't...") The faster you cycle through projects, the faster you'll evolve.

Plain materials have a charm like small scale. And in addition there's the challenge of making do with less. Every designer's ears perk up at the mention of that game, because it's a game you can't lose. Like the JV playing the varsity, if you even tie, you win. So paradoxically there are cases where fewer resources yield better results, because the designers' pleasure at their own ingenuity more than compensates. [5]

So if you're an outsider, take advantage of your ability to make small and inexpensive things. Cultivate the pleasure and simplicity of that kind of work; one day you'll miss it.

Responsibility

When you're old and eminent, what will you miss about being young and obscure? What people seem to miss most is the lack of responsibilities.

Responsibility is an occupational disease of eminence. In principle you could avoid it, just as in principle you could avoid getting fat as you get old, but few do. I sometimes suspect that responsibility is a trap and that the most virtuous route would be to shirk it, but regardless it's certainly constraining.

When you're an outsider you're constrained too, of course. You're short of money, for example. But that constrains you in different ways. How does responsibility constrain you? The worst thing is that it allows you not to focus on real work. Just as the most dangerous forms of procrastination are those that seem like work, the danger of responsibilities is not just that they can consume a whole day, but that they can do it without setting off the kind of alarms you'd set off if you spent a whole day sitting on a park bench.

A lot of the pain of being an outsider is being aware of one's own procrastination. But this is actually a good thing. You're at least close enough to work that the smell of it makes you hungry.

As an outsider, you're just one step away from getting things done. A huge step, admittedly, and one that most people never seem to make, but only one step. If you can summon up the energy to get started, you can work on projects with an intensity (in both senses) that few insiders can match. For insiders work turns into a duty, laden with responsibilities and expectations. It's never so pure as it was when they were young.

Work like a dog being taken for a walk, instead of an ox being yoked to the plow. That's what they miss.

Audience

A lot of outsiders make the mistake of doing the opposite; they admire the eminent so much that they copy even their flaws. Copying is a good way to learn, but copy the right things. When I was in college I imitated the pompous diction of famous professors. But this wasn't what *made* them eminent — it was more a flaw their eminence had allowed them to sink into. Imitating it was like pretending to have gout in order to seem rich.

Half the distinguishing qualities of the eminent are actually disadvantages. Imitating these is not only a waste of time, but will make you seem a fool to your models, who are often well aware of it.

What are the genuine advantages of being an insider? The greatest is an audience. It often seems to outsiders that the great advantage of insiders is money — that they have the resources to do what they want. But so do people who inherit money, and that doesn't seem to help, not as much as an audience. It's good for morale to know people want to see what you're making; it draws work out of you.

If I'm right that the defining advantage of insiders is an audience, then we live in exciting times, because just in the last ten years the Internet has made audiences a lot more liquid. Outsiders don't have to content themselves anymore with a proxy audience of a few smart friends. Now, thanks to the Internet, they can start to grow themselves actual audiences. This is great news for the marginal, who retain the advantages of outsiders while increasingly being able to siphon off what had till recently been the prerogative of the elite.

Though the Web has been around for more than ten years, I think we're just beginning to see its democratizing effects. Outsiders are still learning how to steal audiences. But more importantly, audiences are still learning how to be stolen — they're still just beginning to realize how much deeper bloggers can dig than journalists, how much more interesting a democratic news site can be than a front page controlled by editors, and how much funnier a bunch of kids with webcams can be than mass-produced sitcoms.

The big media companies shouldn't worry that people will post their copyrighted material on YouTube. They should worry that people will post their own stuff on

YouTube, and audiences will watch that instead.

Hacking

If I had to condense the power of the marginal into one sentence it would be: just try hacking something together. That phrase draws in most threads I've mentioned here. Hacking something together means deciding what to do as you're doing it, not a subordinate executing the vision of his boss. It implies the result won't be pretty, because it will be made quickly out of inadequate materials. It may work, but it won't be the sort of thing the eminent would want to put their name on. Something hacked together means something that barely solves the problem, or maybe doesn't solve the problem at all, but another you discovered en route. But that's ok, because the main value of that initial version is not the thing itself, but what it leads to. Insiders who daren't walk through the mud in their nice clothes will never make it to the solid ground on the other side.

The word "try" is an especially valuable component. I disagree here with Yoda, who said there is no try. There is try. It implies there's no punishment if you fail. You're driven by curiosity instead of duty. That means the wind of procrastination will be in your favor: instead of avoiding this work, this will be what you do as a way of avoiding other work. And when you do it, you'll be in a better mood. The more the work depends on imagination, the more that matters, because most people have more ideas when they're happy.

If I could go back and redo my twenties, that would be one thing I'd do more of: just try hacking things together. Like many people that age, I spent a lot of time worrying about what I should do. I also spent some time trying to build stuff. I should have spent less time worrying and more time building. If you're not sure what to do, make something.

Raymond Chandler's advice to thriller writers was "When in doubt, have a man come through a door with a gun in his hand." He followed that advice. Judging from his books, he was often in doubt. But though the result is occasionally cheesy, it's never boring. In life, as in books, action is underrated.

Fortunately the number of things you can just hack together keeps increasing. People fifty years ago would be astonished that one could just hack together a movie, for example. Now you can even hack together distribution. Just make stuff and put it online.

Inappropriate

If you really want to score big, the place to focus is the margin of the margin: the territories only recently captured from the insiders. That's where you'll find the juiciest projects still undone, either because they seemed too risky, or simply because there were too few insiders to explore everything.

This is why I spend most of my time writing essays lately. The writing of essays

used to be limited to those who could get them published. In principle you could have written them and just shown them to your friends; in practice that didn't work. [6] An essayist needs the resistance of an audience, just as an engraver needs the resistance of the plate.

Up till a few years ago, writing essays was the ultimate insider's game. Domain experts were allowed to publish essays about their field, but the pool allowed to write on general topics was about eight people who went to the right parties in New York. Now the reconquista has overrun this territory, and, not surprisingly, found it sparsely cultivated. There are so many essays yet unwritten. They tend to be the naughtier ones; the insiders have pretty much exhausted the motherhood and apple pie topics.

This leads to my final suggestion: a technique for determining when you're on the right track. You're on the right track when people complain that you're unqualified, or that you've done something inappropriate. If people are complaining, that means you're doing something rather than sitting around, which is the first step. And if they're driven to such empty forms of complaint, that means you've probably done something good.

If you make something and people complain that it doesn't *work*, that's a problem. But if the worst thing they can hit you with is your own status as an outsider, that implies that in every other respect you've succeeded. Pointing out that someone is unqualified is as desperate as resorting to racial slurs. It's just a legitimate sounding way of saying: we don't like your type around here.

But the best thing of all is when people call what you're doing inappropriate. I've been hearing this word all my life and I only recently realized that it is, in fact, the sound of the homing beacon. "Inappropriate" is the null criticism. It's merely the adjective form of "I don't like it."

So that, I think, should be the highest goal for the marginal. Be inappropriate. When you hear people saying that, you're golden. And they, incidentally, are busted.

Notes

[1] The facts about Apple's early history are from an interview with Steve Wozniak in Jessica Livingston's *Founders at Work*.

[2] As usual the popular image is several decades behind reality. Now the misunderstood artist is not a chain-smoking drunk who pours his soul into big, messy canvases that philistines see and say "that's not art" because it isn't a picture of anything. The philistines have now been trained that anything hung on a

wall is art. Now the misunderstood artist is a coffee-drinking vegan cartoonist whose work they see and say "that's not art" because it looks like stuff they've seen in the Sunday paper.

[3] In fact this would do fairly well as a definition of politics: what determines rank in the absence of objective tests.

[4] In high school you're led to believe your whole future depends on where you go to college, but it turns out only to buy you a couple years. By your mid-twenties the people worth impressing already judge you more by what you've done than where you went to school.

[5] Managers are presumably wondering, how can I make this miracle happen? How can I make the people working for me do more with less? Unfortunately the constraint probably has to be self-imposed. If you're *expected* to do more with less, then you're being starved, not eating virtuously.

[6] Without the prospect of publication, the closest most people come to writing essays is to write in a journal. I find I never get as deeply into subjects as I do in proper essays. As the name implies, you don't go back and rewrite journal entries over and over for two weeks.

Thanks to Sam Altman, Trevor Blackwell, Paul Buchheit, Sarah Harlin, Jessica Livingston, Jackie McDonough, Robert Morris, Olin Shivers, and Chris Small for reading drafts of this, and to Chris Small and Chad Fowler for inviting me to speak.

Japanese Translation

Chinese Translation

The Island Test

July 2006

I've discovered a handy test for figuring out what you're addicted to. Imagine you were going to spend the weekend at a friend's house on a little island off the coast of Maine. There are no shops on the island and you won't be able to leave while you're there. Also, you've never been to this house before, so you can't assume it will have more than any house might.

What, besides clothes and toiletries, do you make a point of packing? That's what you're addicted to. For example, if you find yourself packing a bottle of vodka (just in case), you may want to stop and think about that.

For me the list is four things: books, earplugs, a notebook, and a pen.

There are other things I might bring if I thought of it, like music, or tea, but I can live without them. I'm not so addicted to caffeine that I wouldn't risk the house not having any tea, just for a weekend.

Quiet is another matter. I realize it seems a bit eccentric to take earplugs on a trip to an island off the coast of Maine. If anywhere should be quiet, that should. But what if the person in the next room snored? What if there was a kid playing basketball? (Thump, thump, thump... thump.) Why risk it? Earplugs are small.

Sometimes I can think with noise. If I already have momentum on some project, I can work in noisy places. I can edit an essay or debug code in an airport. But airports are not so bad: most of the noise is whitish. I couldn't work with the sound of a sitcom coming through the wall, or a car in the street playing thump-thump music.

And of course there's another kind of thinking, when you're starting something new, that requires complete quiet. You never know when this will strike. It's just as well to carry plugs.

The notebook and pen are professional equipment, as it were. Though actually there is something druglike about them, in the sense that their main purpose is to make me feel better. I hardly ever go back and read stuff I write down in notebooks. It's just that if I can't write things down, worrying about remembering one idea gets in the way of having the next. Pen and paper wick ideas.

The best notebooks I've found are made by a company called Miquelrius. I use their smallest size, which is about 2.5 x 4 in. The secret to writing on such narrow pages is to break words only when you run out of space, like a Latin inscription. I use the cheapest plastic Bic ballpoints, partly because their gluey ink doesn't seep through pages, and partly so I don't worry about losing them.

I only started carrying a notebook about three years ago. Before that I used whatever scraps of paper I could find. But the problem with scraps of paper is that they're not ordered. In a notebook you can guess what a scribble means by looking at the pages around it. In the scrap era I was constantly finding notes I'd written years before that might say something I needed to remember, if I could only figure out what.

As for books, I know the house would probably have something to read. On the average trip I bring four books and only read one of them, because I find new books to read en route. Really bringing books is insurance.

I realize this dependence on books is not entirely good—that what I need them for is distraction. The books I bring on trips are often quite virtuous, the sort of stuff that might be assigned reading in a college class. But I know my motives aren't virtuous. I bring books because if the world gets boring I need to be able to slip into another distilled by some writer. It's like eating jam when you know you should be eating fruit.

There is a point where I'll do without books. I was walking in some steep mountains once, and decided I'd rather just think, if I was bored, rather than carry a single unnecessary ounce. It wasn't so bad. I found I could entertain myself by having ideas instead of reading other people's. If you stop eating jam, fruit starts to taste better.

So maybe I'll try not bringing books on some future trip. They're going to have to pry the plugs out of my cold, dead ears, however.

[Spanish Translation]

[Japanese Translation]

Copy What You Like

July 2006

When I was in high school I spent a lot of time imitating bad writers. What we studied in English classes was mostly fiction, so I assumed that was the highest form of writing. Mistake number one. The stories that seemed to be most admired were ones in which people suffered in complicated ways. Anything funny or gripping was ipso facto suspect, unless it was old enough to be hard to understand, like Shakespeare or Chaucer. Mistake number two. The ideal medium seemed the short story, which I've since learned had quite a brief life, roughly coincident with the peak of magazine publishing. But since their size made them perfect for use in high school classes, we read a lot of them, which gave us the impression the short story was flourishing. Mistake number three. And because they were so short, nothing really had to happen; you could just show a randomly truncated slice of life, and that was considered advanced. Mistake number four. The result was that I wrote a lot of stories in which nothing happened except that someone was unhappy in a way that seemed deep.

For most of college I was a philosophy major. I was very impressed by the papers published in philosophy journals. They were so beautifully typeset, and their tone was just captivating—alternately casual and buffer-overflowingly technical. A fellow would be walking along a street and suddenly modality qua modality would spring upon him. I didn't ever quite understand these papers, but I figured I'd get around to that later, when I had time to reread them more closely. In the meantime I tried my best to imitate them. This was, I can now see, a doomed undertaking, because they weren't really saying anything. No philosopher ever refuted another, for

example, because no one said anything definite enough to refute. Needless to say, my imitations didn't say anything either.

In grad school I was still wasting time imitating the wrong things. There was then a fashionable type of program called an expert system, at the core of which was something called an inference engine. I looked at what these things did and thought "I could write that in a thousand lines of code." And yet eminent professors were writing books about them, and startups were selling them for a year's salary a copy. What an opportunity, I thought; these impressive things seem easy to me; I must be pretty sharp. Wrong. It was simply a fad. The books the professors wrote about expert systems are now ignored. They were not even on a *path* to anything interesting. And the customers paying so much for them were largely the same government agencies that paid thousands for screwdrivers and toilet seats.

How do you avoid copying the wrong things? Copy only what you genuinely like. That would have saved me in all three cases. I didn't enjoy the short stories we had to read in English classes; I didn't learn anything from philosophy papers; I didn't use expert systems myself. I believed these things were good because they were admired.

It can be hard to separate the things you like from the things you're impressed with. One trick is to ignore presentation. Whenever I see a painting impressively hung in a museum, I ask myself: how much would I pay for this if I found it at a garage sale, dirty and frameless, and with no idea who painted it? If you walk around a museum trying this experiment, you'll find you get some truly startling results. Don't ignore this data point just because it's an outlier.

Another way to figure out what you like is to look at what you enjoy as guilty pleasures. Many things people like, especially if they're young and ambitious, they like largely for the feeling of virtue in liking them. 99% of people reading *Ulysses* are thinking "I'm reading *Ulysses*" as they do it. A guilty pleasure is at least a pure one. What do you read when you don't feel up to being virtuous? What kind of book do you read and feel sad that there's only half of it left, instead of being impressed that you're half way through? That's what you really like.

Even when you find genuinely good things to copy, there's another pitfall to be avoided. Be careful to copy what makes them good, rather than their flaws. It's easy to be drawn into imitating flaws, because they're easier to see, and of course easier to copy too. For example, most painters in the eighteenth and nineteenth centuries used brownish colors. They were imitating the great painters of the Renaissance, whose paintings by that time were brown with dirt. Those paintings have since been cleaned, revealing brilliant colors; their imitators are of course still brown.

It was painting, incidentally, that cured me of copying the wrong things. Halfway through grad school I decided I wanted to try being a painter, and the art world was so manifestly corrupt that it snapped the leash of credulity. These people

made philosophy professors seem as scrupulous as mathematicians. It was so clearly a choice of doing good work xor being an insider that I was forced to see the distinction. It's there to some degree in almost every field, but I had till then managed to avoid facing it.

That was one of the most valuable things I learned from painting: you have to figure out for yourself what's good. You can't trust authorities. They'll lie to you on this one.

 Comment on this essay.

Chinese Translation

Romanian Translation

Spanish Translation

Russian Translation

How to Present to Investors

August 2006, rev. April 2007, September 2010

In a few days it will be Demo Day, when the startups we funded this summer present to investors. Y Combinator funds startups twice a year, in January and June. Ten weeks later we invite all the investors we know to hear them present what they've built so far.

Ten weeks is not much time. The average startup probably doesn't have much to show for itself after ten weeks. But the average startup fails. When you look at the ones that went on to do great things, you find a lot that began with someone pounding out a prototype in a week or two of nonstop work. Startups are a counterexample to the rule that haste makes waste.

(Too much money seems to be as bad for startups as too much time, so we don't give them much money either.)

A week before Demo Day, we have a dress rehearsal called Rehearsal Day. At other Y Combinator events we allow outside guests, but not at Rehearsal Day. No one except the other founders gets to see the rehearsals.

The presentations on Rehearsal Day are often pretty rough. But this is to be expected. We try to pick founders who are good at building things, not ones who are slick presenters. Some of the founders are just out of college, or even still in it, and have never spoken to a group of people they didn't already know.

So we concentrate on the basics. On Demo Day each startup will only get ten minutes, so we encourage them to focus on just two goals: (a) explain what you're doing, and (b) explain why users will want it.

That might sound easy, but it's not when the speakers have no experience presenting, and they're explaining technical matters to an audience that's mostly non-technical.

This situation is constantly repeated when startups present to investors: people who are bad at explaining, talking to people who are bad at understanding. Practically every successful startup, including stars like Google, presented at some point to investors who didn't get it and turned them down. Was it because the founders were bad at presenting, or because the investors were obtuse? It's probably always some of both.

At the most recent Rehearsal Day, we four Y Combinator partners found ourselves saying a lot of the same things we said at the last two. So at dinner afterward we collected all our tips about presenting to investors. Most startups face similar challenges, so we hope these will be useful to a wider audience.

1. Explain what you're doing.

Investors' main question when judging a very early startup is whether you've made a compelling product. Before they can judge whether you've built a good x, they have to understand what kind of x you've built. They will get very frustrated if instead of telling them what you do, you make them sit through some kind of preamble.

Say what you're doing as soon as possible, preferably in the first sentence. "We're Jeff and Bob and we've built an easy to use web-based database. Now we'll show it to you and explain why people need this."

If you're a great public speaker you may be able to violate this rule. Last year one founder spent the whole first half of his talk on a fascinating analysis of the limits of the conventional desktop metaphor. He got away with it, but unless you're a captivating speaker, which most hackers aren't, it's better to play it safe.

2. Get rapidly to demo.

This section is now obsolete for YC founders presenting at Demo Day, because Demo Day presentations are now so short that they rarely include much if any demo. They seem to work just as well without, however, which makes me think I was wrong to emphasize demos so much before.

A demo explains what you've made more effectively than any verbal description. The only thing worth talking about first is the problem you're trying to solve and why it's important. But don't spend more than a tenth of your time on that. Then demo.

When you demo, don't run through a catalog of features. Instead start with the problem you're solving, and then show how your product solves it. Show features

in an order driven by some kind of purpose, rather than the order in which they happen to appear on the screen.

If you're demoing something web-based, assume that the network connection will mysteriously die 30 seconds into your presentation, and come prepared with a copy of the server software running on your laptop.

3. Better a narrow description than a vague one.

One reason founders resist describing their projects concisely is that, at this early stage, there are all kinds of possibilities. The most concise descriptions seem misleadingly narrow. So for example a group that has built an easy web-based database might resist calling their applicaton that, because it could be so much more. In fact, it could be anything...

The problem is, as you approach (in the calculus sense) a description of something that could be anything, the content of your description approaches zero. If you describe your web-based database as "a system to allow people to collaboratively leverage the value of information," it will go in one investor ear and out the other. They'll just discard that sentence as meaningless boilerplate, and hope, with increasing impatience, that in the next sentence you'll actually explain what you've made.

Your primary goal is not to describe everything your system might one day become, but simply to convince investors you're worth talking to further. So approach this like an algorithm that gets the right answer by successive approximations. Begin with a description that's gripping but perhaps overly narrow, then flesh it out to the extent you can. It's the same principle as incremental development: start with a simple prototype, then add features, but at every point have working code. In this case, "working code" means a working description in the investor's head.

4. Don't talk and drive.

Have one person talk while another uses the computer. If the same person does both, they'll inevitably mumble downwards at the computer screen instead of talking clearly at the audience.

As long as you're standing near the audience and looking at them, politeness (and habit) compel them to pay attention to you. Once you stop looking at them to fuss with something on your computer, their minds drift off to the errands they have to run later.

5. Don't talk about secondary matters at length.

If you only have a few minutes, spend them explaining what your product does and why it's great. Second order issues like competitors or resumes should be single slides you go through quickly at the end. If you have impressive resumes,

just flash them on the screen for 15 seconds and say a few words. For competitors, list the top 3 and explain in one sentence each what they lack that you have. And put this kind of thing at the end, after you've made it clear what you've built.

6. Don't get too deeply into business models.

It's good to talk about how you plan to make money, but mainly because it shows you care about that and have thought about it. Don't go into detail about your business model, because (a) that's not what smart investors care about in a brief presentation, and (b) any business model you have at this point is probably wrong anyway.

Recently a VC who came to speak at Y Combinator talked about a company he just invested in. He said their business model was wrong and would probably change three times before they got it right. The founders were experienced guys who'd done startups before and who'd just succeeded in getting millions from one of the top VC firms, and even their business model was crap. (And yet he invested anyway, because he expected it to be crap at this stage.)

If you're solving an important problem, you're going to sound a lot smarter talking about that than the business model. The business model is just a bunch of guesses, and guesses about stuff that's probably not your area of expertise. So don't spend your precious few minutes talking about crap when you could be talking about solid, interesting things you know a lot about: the problem you're solving and what you've built so far.

As well as being a bad use of time, if your business model seems spectacularly wrong, that will push the stuff you want investors to remember out of their heads. They'll just remember you as the company with the boneheaded plan for making money, rather than the company that solved that important problem.

7. Talk slowly and clearly at the audience.

Everyone at Rehearsal Day could see the difference between the people who'd been out in the world for a while and had presented to groups, and those who hadn't.

You need to use a completely different voice and manner talking to a roomful of people than you would in conversation. Everyday life gives you no practice in this. If you can't already do it, the best solution is to treat it as a consciously artificial trick, like juggling.

However, that doesn't mean you should talk like some kind of announcer. Audiences tune that out. What you need to do is talk in this artificial way, and yet make it seem conversational. (Writing is the same. Good writing is an elaborate effort to seem spontaneous.)

If you want to write out your whole presentation beforehand and memorize it,

that's ok. That has worked for some groups in the past. But make sure to write something that sounds like spontaneous, informal speech, and deliver it that way too.

Err on the side of speaking slowly. At Rehearsal Day, one of the founders mentioned a rule actors use: if you feel you're speaking too slowly, you're speaking at about the right speed.

8. Have one person talk.

Startups often want to show that all the founders are equal partners. This is a good instinct; investors dislike unbalanced teams. But trying to show it by partitioning the presentation is going too far. It's distracting. You can demonstrate your respect for one another in more subtle ways. For example, when one of the groups presented at Demo Day, the more extroverted of the two founders did most of the talking, but he described his co-founder as the best hacker he'd ever met, and you could tell he meant it.

Pick the one or at most two best speakers, and have them do most of the talking.

Exception: If one of the founders is an expert in some specific technical field, it can be good for them to talk about that for a minute or so. This kind of "expert witness" can add credibility, even if the audience doesn't understand all the details. If Jobs and Wozniak had 10 minutes to present the Apple II, it might be a good plan to have Jobs speak for 9 minutes and have Woz speak for a minute in the middle about some of the technical feats he'd pulled off in the design. (Though of course if it were actually those two, Jobs would speak for the entire 10 minutes.)

9. Seem confident.

Between the brief time available and their lack of technical background, many in the audience will have a hard time evaluating what you're doing. Probably the single biggest piece of evidence, initially, will be your own confidence in it. You have to show you're impressed with what you've made.

And I mean show, not tell. Never say "we're passionate" or "our product is great." People just ignore that—or worse, write you off as bullshitters. Such messages must be implicit.

What you must not do is seem nervous and apologetic. If you've truly made something good, you're doing investors a *favor* by telling them about it. If you don't genuinely believe that, perhaps you ought to change what your company is doing. If you don't believe your startup has such promise that you'd be doing them a favor by letting them invest, why are you investing your time in it?

10. Don't try to seem more than you are.

Don't worry if your company is just a few months old and doesn't have an office

yet, or your founders are technical people with no business experience. Google was like that once, and they turned out ok. Smart investors can see past such superficial flaws. They're not looking for finished, smooth presentations. They're looking for raw talent. All you need to convince them of is that you're smart and that you're onto something good. If you try too hard to conceal your rawness—by trying to seem corporate, or pretending to know about stuff you don't—you may just conceal your talent.

You can afford to be candid about what you haven't figured out yet. Don't go out of your way to bring it up (e.g. by having a slide about what might go wrong), but don't try to pretend either that you're further along than you are. If you're a hacker and you're presenting to experienced investors, they're probably better at detecting bullshit than you are at producing it.

11. Don't put too many words on slides.

When there are a lot of words on a slide, people just skip reading it. So look at your slides and ask of each word "could I cross this out?" This includes gratuitous clip art. Try to get your slides under 20 words if you can.

Don't read your slides. They should be something in the background as you face the audience and talk to them, not something you face and read to an audience sitting behind you.

Cluttered sites don't do well in demos, especially when they're projected onto a screen. At the very least, crank up the font size big enough to make all the text legible. But cluttered sites are bad anyway, so perhaps you should use this opportunity to make your design simpler.

12. Specific numbers are good.

If you have any kind of data, however preliminary, tell the audience. Numbers stick in people's heads. If you can claim that the median visitor generates 12 page views, that's great.

But don't give them more than four or five numbers, and only give them numbers specific to you. You don't need to tell them the size of the market you're in. Who cares, really, if it's 500 million or 5 billion a year? Talking about that is like an actor at the beginning of his career telling his parents how much Tom Hanks makes. Yeah, sure, but first you have to become Tom Hanks. The important part is not whether he makes ten million a year or a hundred, but how you get there.

13. Tell stories about users.

The biggest fear of investors looking at early stage startups is that you've built something based on your own a priori theories of what the world needs, but that no one will actually want. So it's good if you can talk about problems specific users have and how you solve them.

Greg Mcadoo said one thing Sequoia looks for is the "proxy for demand." What are people doing now, using inadequate tools, that shows they need what you're making?

Another sign of user need is when people pay a lot for something. It's easy to convince investors there will be demand for a cheaper alternative to something popular, if you preserve the qualities that made it popular.

The best stories about user needs are about your own. A remarkable number of famous startups grew out of some need the founders had: Apple, Microsoft, Yahoo, Google. Experienced investors know that, so stories of this type will get their attention. The next best thing is to talk about the needs of people you know personally, like your friends or siblings.

14. Make a soundbite stick in their heads.

Professional investors hear a lot of pitches. After a while they all blur together. The first cut is simply to be one of those they remember. And the way to ensure that is to create a descriptive phrase about yourself that sticks in their heads.

In Hollywood, these phrases seem to be of the form "x meets y." In the startup world, they're usually "the x of y" or "the x y." Viaweb's was "the Microsoft Word of ecommerce."

Find one and launch it clearly (but apparently casually) in your talk, preferably near the beginning.

It's a good exercise for you, too, to sit down and try to figure out how to describe your startup in one compelling phrase. If you can't, your plans may not be sufficiently focused.

[How to Fund a Startup]

[Hackers' Guide to Investors]

[Spanish Translation]

[Japanese Translation]

[Russian Translation]

Image: Casey Muller: Trevor Blackwell at Rehearsal Day, summer 2006

A Student's Guide to Startups

October 2006

(This essay is derived from a talk at MIT.)

Till recently graduating seniors had two choices: get a job or go to grad school. I think there will increasingly be a third option: to start your own startup. But how common will that be?

I'm sure the default will always be to get a job, but starting a startup could well become as popular as grad school. In the late 90s my professor friends used to complain that they couldn't get grad students, because all the undergrads were going to work for startups. I wouldn't be surprised if that situation returns, but with one difference: this time they'll be starting their own instead of going to work for other people's.

The most ambitious students will at this point be asking: Why wait till you graduate? Why not start a startup while you're in college? In fact, why go to college at all? Why not start a startup instead?

A year and a half ago I gave a talk where I said that the average age of the founders of Yahoo, Google, and Microsoft was 24, and that if grad students could start startups, why not undergrads? I'm glad I phrased that as a question, because now I can pretend it wasn't merely a rhetorical one. At the time I couldn't imagine why there should be any lower limit for the age of startup founders. Graduation is a bureaucratic change, not a biological one. And certainly there are undergrads as competent technically as most grad students. So why shouldn't undergrads be able to start startups as well as grad students?

I now realize that something does change at graduation: you lose a huge excuse for failing. Regardless of how complex your life is, you'll find that everyone else, including your family and friends, will discard all the low bits and regard you as having a single occupation at any given time. If you're in college and have a summer job writing software, you still read as a student. Whereas if you graduate and get a job programming, you'll be instantly regarded by everyone as a programmer.

The problem with starting a startup while you're still in school is that there's a built-in escape hatch. If you start a startup in the summer between your junior and senior year, it reads to everyone as a summer job. So if it goes nowhere, big deal; you return to school in the fall with all the other seniors; no one regards you as a failure, because your occupation is student, and you didn't fail at that. Whereas if you start a startup just one year later, after you graduate, as long as you're not

accepted to grad school in the fall the startup reads to everyone as your occupation. You're now a startup founder, so you have to do well at that.

For nearly everyone, the opinion of one's peers is the most powerful motivator of all—more powerful even than the nominal goal of most startup founders, getting rich. [1] About a month into each funding cycle we have an event called Prototype Day where each startup presents to the others what they've got so far. You might think they wouldn't need any more motivation. They're working on their cool new idea; they have funding for the immediate future; and they're playing a game with only two outcomes: wealth or failure. You'd think that would be motivation enough. And yet the prospect of a demo pushes most of them into a rush of activity.

Even if you start a startup explicitly to get rich, the money you might get seems pretty theoretical most of the time. What drives you day to day is not wanting to look bad.

You probably can't change that. Even if you could, I don't think you'd want to; someone who really, truly doesn't care what his peers think of him is probably a psychopath. So the best you can do is consider this force like a wind, and set up your boat accordingly. If you know your peers are going to push you in some direction, choose good peers, and position yourself so they push you in a direction you like.

Graduation changes the prevailing winds, and those make a difference. Starting a startup is so hard that it's a close call even for the ones that succeed. However high a startup may be flying now, it probably has a few leaves stuck in the landing gear from those trees it barely cleared at the end of the runway. In such a close game, the smallest increase in the forces against you can be enough to flick you over the edge into failure.

When we first started Y Combinator we encouraged people to start startups while they were still in college. That's partly because Y Combinator began as a kind of summer program. We've kept the program shape—all of us having dinner together once a week turns out to be a good idea—but we've decided now that the party line should be to tell people to wait till they graduate.

Does that mean you can't start a startup in college? Not at all. Sam Altman, the co-founder of Loopt, had just finished his sophomore year when we funded them, and Loopt is probably the most promising of all the startups we've funded so far. But Sam Altman is a very unusual guy. Within about three minutes of meeting him, I remember thinking "Ah, so this is what Bill Gates must have been like when he was 19."

If it can work to start a startup during college, why do we tell people not to? For the same reason that the probably apocryphal violinist, whenever he was asked to judge someone's playing, would always say they didn't have enough talent to make it as a pro. Succeeding as a musician takes determination as well as talent, so this answer works out to be the right advice for everyone. The ones who are uncertain believe it and give up, and the ones who are sufficiently determined think "screw that, I'll succeed anyway."

So our official policy now is only to fund undergrads we can't talk out of it. And frankly, if you're not certain, you *should* wait. It's not as if all the opportunities to start companies are going to be gone if you don't do it now. Maybe the window will

close on some idea you're working on, but that won't be the last idea you'll have. For every idea that times out, new ones become feasible. Historically the opportunities to start startups have only increased with time.

In that case, you might ask, why not wait longer? Why not go work for a while, or go to grad school, and then start a startup? And indeed, that might be a good idea. If I had to pick the sweet spot for startup founders, based on who we're most excited to see applications from, I'd say it's probably the mid-twenties. Why? What advantages does someone in their mid-twenties have over someone who's 21? And why isn't it older? What can 25 year olds do that 32 year olds can't? Those turn out to be questions worth examining.

Plus

If you start a startup soon after college, you'll be a young founder by present standards, so you should know what the relative advantages of young founders are. They're not what you might think. As a young founder your strengths are: stamina, poverty, rootlessness, colleagues, and ignorance.

The importance of stamina shouldn't be surprising. If you've heard anything about startups you've probably heard about the long hours. As far as I can tell these are universal. I can't think of any successful startups whose founders worked 9 to 5. And it's particularly necessary for younger founders to work long hours because they're probably not as efficient as they'll be later.

Your second advantage, poverty, might not sound like an advantage, but it is a huge one. Poverty implies you can live cheaply, and this is critically important for startups. Nearly every startup that fails, fails by running out of money. It's a little misleading to put it this way, because there's usually some other underlying cause. But regardless of the source of your problems, a low burn rate gives you more opportunity to recover from them. And since most startups make all kinds of mistakes at first, room to recover from mistakes is a valuable thing to have.

Most startups end up doing something different than they planned. The way the successful ones find something that works is by trying things that don't. So the worst thing you can do in a startup is to have a rigid, pre-ordained plan and then start spending a lot of money to implement it. Better to operate cheaply and give your ideas time to evolve.

Recent grads can live on practically nothing, and this gives you an edge over older founders, because the main cost in software startups is people. The guys with kids and mortgages are at a real disadvantage. This is one reason I'd bet on the 25 year old over the 32 year old. The 32 year old probably is a better programmer, but probably also has a much more expensive life. Whereas a 25 year old has some work experience (more on that later) but can live as cheaply as an undergrad.

Robert Morris and I were 29 and 30 respectively when we started Viaweb, but fortunately we still lived like 23 year olds. We both had roughly zero assets. I would have loved to have a mortgage, since that would have meant I had a *house*. But in retrospect having nothing turned out to be convenient. I wasn't tied down and I was used to living cheaply.

Even more important than living cheaply, though, is thinking cheaply. One reason

the Apple II was so popular was that it was cheap. The computer itself was cheap, and it used cheap, off-the-shelf peripherals like a cassette tape recorder for data storage and a TV as a monitor. And you know why? Because Woz designed this computer for himself, and he couldn't afford anything more.

We benefitted from the same phenomenon. Our prices were daringly low for the time. The top level of service was $300 a month, which was an order of magnitude below the norm. In retrospect this was a smart move, but we didn't do it because we were smart. $300 a month seemed like a lot of money to us. Like Apple, we created something inexpensive, and therefore popular, simply because we were poor.

A lot of startups have that form: someone comes along and makes something for a tenth or a hundredth of what it used to cost, and the existing players can't follow because they don't even want to think about a world in which that's possible. Traditional long distance carriers, for example, didn't even want to think about VoIP. (It was coming, all the same.) Being poor helps in this game, because your own personal bias points in the same direction technology evolves in.

The advantages of rootlessness are similar to those of poverty. When you're young you're more mobile—not just because you don't have a house or much stuff, but also because you're less likely to have serious relationships. This turns out to be important, because a lot of startups involve someone moving.

The founders of Kiko, for example, are now en route to the Bay Area to start their next startup. It's a better place for what they want to do. And it was easy for them to decide to go, because neither as far as I know has a serious girlfriend, and everything they own will fit in one car—or more precisely, will either fit in one car or is crappy enough that they don't mind leaving it behind.

They at least were in Boston. What if they'd been in Nebraska, like Evan Williams was at their age? Someone wrote recently that the drawback of Y Combinator was that you had to move to participate. It couldn't be any other way. The kind of conversations we have with founders, we have to have in person. We fund a dozen startups at a time, and we can't be in a dozen places at once. But even if we could somehow magically save people from moving, we wouldn't. We wouldn't be doing founders a favor by letting them stay in Nebraska. Places that aren't startup hubs are toxic to startups. You can tell that from indirect evidence. You can tell how hard it must be to start a startup in Houston or Chicago or Miami from the microscopically small number, per capita, that succeed there. I don't know exactly what's suppressing all the startups in these towns—probably a hundred subtle little things—but something must be. [2]

Maybe this will change. Maybe the increasing cheapness of startups will mean they'll be able to survive anywhere, instead of only in the most hospitable environments. Maybe 37signals is the pattern for the future. But maybe not. Historically there have always been certain towns that were centers for certain industries, and if you weren't in one of them you were at a disadvantage. So my guess is that 37signals is an anomaly. We're looking at a pattern much older than "Web 2.0" here.

Perhaps the reason more startups per capita happen in the Bay Area than Miami is simply that there are more founder-type people there. Successful startups are almost never started by one person. Usually they begin with a conversation in

which someone mentions that something would be a good idea for a company, and his friend says, "Yeah, that is a good idea, let's try it." If you're missing that second person who says "let's try it," the startup never happens. And that is another area where undergrads have an edge. They're surrounded by people willing to say that. At a good college you're concentrated together with a lot of other ambitious and technically minded people—probably more concentrated than you'll ever be again. If your nucleus spits out a neutron, there's a good chance it will hit another nucleus.

The number one question people ask us at Y Combinator is: Where can I find a co-founder? That's the biggest problem for someone starting a startup at 30. When they were in school they knew a lot of good co-founders, but by 30 they've either lost touch with them or these people are tied down by jobs they don't want to leave.

Viaweb was an anomaly in this respect too. Though we were comparatively old, we weren't tied down by impressive jobs. I was trying to be an artist, which is not very constraining, and Robert, though 29, was still in grad school due to a little interruption in his academic career back in 1988. So arguably the Worm made Viaweb possible. Otherwise Robert would have been a junior professor at that age, and he wouldn't have had time to work on crazy speculative projects with me.

Most of the questions people ask Y Combinator we have some kind of answer for, but not the co-founder question. There is no good answer. Co-founders really should be people you already know. And by far the best place to meet them is school. You have a large sample of smart people; you get to compare how they all perform on identical tasks; and everyone's life is pretty fluid. A lot of startups grow out of schools for this reason. Google, Yahoo, and Microsoft, among others, were all founded by people who met in school. (In Microsoft's case, it was high school.)

Many students feel they should wait and get a little more experience before they start a company. All other things being equal, they should. But all other things are not quite as equal as they look. Most students don't realize how rich they are in the scarcest ingredient in startups, co-founders. If you wait too long, you may find that your friends are now involved in some project they don't want to abandon. The better they are, the more likely this is to happen.

One way to mitigate this problem might be to actively plan your startup while you're getting those n years of experience. Sure, go off and get jobs or go to grad school or whatever, but get together regularly to scheme, so the idea of starting a startup stays alive in everyone's brain. I don't know if this works, but it can't hurt to try.

It would be helpful just to realize what an advantage you have as students. Some of your classmates are probably going to be successful startup founders; at a great technical university, that is a near certainty. So which ones? If I were you I'd look for the people who are not just smart, but incurable builders. Look for the people who keep starting projects, and finish at least some of them. That's what we look for. Above all else, above academic credentials and even the idea you apply with, we look for people who build things.

The other place co-founders meet is at work. Fewer do than at school, but there are things you can do to improve the odds. The most important, obviously, is to work somewhere that has a lot of smart, young people. Another is to work for a

company located in a startup hub. It will be easier to talk a co-worker into quitting with you in a place where startups are happening all around you.

You might also want to look at the employment agreement you sign when you get hired. Most will say that any ideas you think of while you're employed by the company belong to them. In practice it's hard for anyone to prove what ideas you had when, so the line gets drawn at code. If you're going to start a startup, don't write any of the code while you're still employed. Or at least discard any code you wrote while still employed and start over. It's not so much that your employer will find out and sue you. It won't come to that; investors or acquirers or (if you're so lucky) underwriters will nail you first. Between t = 0 and when you buy that yacht, *someone* is going to ask if any of your code legally belongs to anyone else, and you need to be able to say no. [3]

The most overreaching employee agreement I've seen so far is Amazon's. In addition to the usual clauses about owning your ideas, you also can't be a founder of a startup that has another founder who worked at Amazon—even if you didn't know them or even work there at the same time. I suspect they'd have a hard time enforcing this, but it's a bad sign they even try. There are plenty of other places to work; you may as well choose one that keeps more of your options open.

Speaking of cool places to work, there is of course Google. But I notice something slightly frightening about Google: zero startups come out of there. In that respect it's a black hole. People seem to like working at Google too much to leave. So if you hope to start a startup one day, the evidence so far suggests you shouldn't work there.

I realize this seems odd advice. If they make your life so good that you don't want to leave, why not work there? Because, in effect, you're probably getting a local maximum. You need a certain activation energy to start a startup. So an employer who's fairly pleasant to work for can lull you into staying indefinitely, even if it would be a net win for you to leave. [4]

The best place to work, if you want to start a startup, is probably a startup. In addition to being the right sort of experience, one way or another it will be over quickly. You'll either end up rich, in which case problem solved, or the startup will get bought, in which case it it will start to suck to work there and it will be easy to leave, or most likely, the thing will blow up and you'll be free again.

Your final advantage, ignorance, may not sound very useful. I deliberately used a controversial word for it; you might equally call it innocence. But it seems to be a powerful force. My Y Combinator co-founder Jessica Livingston is just about to publish a book of interviews with startup founders, and I noticed a remarkable pattern in them. One after another said that if they'd known how hard it would be, they would have been too intimidated to start.

Ignorance can be useful when it's a counterweight to other forms of stupidity. It's useful in starting startups because you're capable of more than you realize. Starting startups is harder than you expect, but you're also capable of more than you expect, so they balance out.

Most people look at a company like Apple and think, how could I ever make such a thing? Apple is an institution, and I'm just a person. But every institution was at one point just a handful of people in a room deciding to start something.

Institutions are made up, and made up by people no different from you.

I'm not saying everyone could start a startup. I'm sure most people couldn't; I don't know much about the population at large. When you get to groups I know well, like hackers, I can say more precisely. At the top schools, I'd guess as many as a quarter of the CS majors could make it as startup founders if they wanted.

That "if they wanted" is an important qualification—so important that it's almost cheating to append it like that—because once you get over a certain threshold of intelligence, which most CS majors at top schools are past, the deciding factor in whether you succeed as a founder is how much you want to. You don't have to be that smart. If you're not a genius, just start a startup in some unsexy field where you'll have less competition, like software for human resources departments. I picked that example at random, but I feel safe in predicting that whatever they have now, it wouldn't take genius to do better. There are a lot of people out there working on boring stuff who are desperately in need of better software, so however short you think you fall of Larry and Sergey, you can ratchet down the coolness of the idea far enough to compensate.

As well as preventing you from being intimidated, ignorance can sometimes help you discover new ideas. Steve Wozniak put this very strongly:

> All the best things that I did at Apple came from (a) not having money and (b) not having done it before, ever. Every single thing that we came out with that was really great, I'd never once done that thing in my life.

When you know nothing, you have to reinvent stuff for yourself, and if you're smart your reinventions may be better than what preceded them. This is especially true in fields where the rules change. All our ideas about software were developed in a time when processors were slow, and memories and disks were tiny. Who knows what obsolete assumptions are embedded in the conventional wisdom? And the way these assumptions are going to get fixed is not by explicitly deallocating them, but by something more akin to garbage collection. Someone ignorant but smart will come along and reinvent everything, and in the process simply fail to reproduce certain existing ideas.

Minus

So much for the advantages of young founders. What about the disadvantages? I'm going to start with what goes wrong and try to trace it back to the root causes.

What goes wrong with young founders is that they build stuff that looks like class projects. It was only recently that we figured this out ourselves. We noticed a lot of similarities between the startups that seemed to be falling behind, but we couldn't figure out how to put it into words. Then finally we realized what it was: they were building class projects.

But what does that really mean? What's wrong with class projects? What's the difference between a class project and a real startup? If we could answer that question it would be useful not just to would-be startup founders but to students in general, because we'd be a long way toward explaining the mystery of the so-

called real world.

There seem to be two big things missing in class projects: (1) an iterative definition of a real problem and (2) intensity.

The first is probably unavoidable. Class projects will inevitably solve fake problems. For one thing, real problems are rare and valuable. If a professor wanted to have students solve real problems, he'd face the same paradox as someone trying to give an example of whatever "paradigm" might succeed the Standard Model of physics. There may well be something that does, but if you could think of an example you'd be entitled to the Nobel Prize. Similarly, good new problems are not to be had for the asking.

In technology the difficulty is compounded by the fact that real startups tend to discover the problem they're solving by a process of evolution. Someone has an idea for something; they build it; and in doing so (and probably only by doing so) they realize the problem they should be solving is another one. Even if the professor let you change your project description on the fly, there isn't time enough to do that in a college class, or a market to supply evolutionary pressures. So class projects are mostly about implementation, which is the least of your problems in a startup.

It's not just that in a startup you work on the idea as well as implementation. The very implementation is different. Its main purpose is to refine the idea. Often the only value of most of the stuff you build in the first six months is that it proves your initial idea was mistaken. And that's extremely valuable. If you're free of a misconception that everyone else still shares, you're in a powerful position. But you're not thinking that way about a class project. Proving your initial plan was mistaken would just get you a bad grade. Instead of building stuff to throw away, you tend to want every line of code to go toward that final goal of showing you did a lot of work.

That leads to our second difference: the way class projects are measured. Professors will tend to judge you by the distance between the starting point and where you are now. If someone has achieved a lot, they should get a good grade. But customers will judge you from the other direction: the distance remaining between where you are now and the features they need. The market doesn't give a shit how hard you worked. Users just want your software to do what they need, and you get a zero otherwise. That is one of the most distinctive differences between school and the real world: there is no reward for putting in a good effort. In fact, the whole concept of a "good effort" is a fake idea adults invented to encourage kids. It is not found in nature.

Such lies seem to be helpful to kids. But unfortunately when you graduate they don't give you a list of all the lies they told you during your education. You have to get them beaten out of you by contact with the real world. And this is why so many jobs want work experience. I couldn't understand that when I was in college. I knew how to program. In fact, I could tell I knew how to program better than

most people doing it for a living. So what was this mysterious "work experience" and why did I need it?

Now I know what it is, and part of the confusion is grammatical. Describing it as "work experience" implies it's like experience operating a certain kind of machine, or using a certain programming language. But really what work experience refers to is not some specific expertise, but the elimination of certain habits left over from childhood.

One of the defining qualities of kids is that they flake. When you're a kid and you face some hard test, you can cry and say "I can't" and they won't make you do it. Of course, no one can make you do anything in the grownup world either. What they do instead is fire you. And when motivated by that you find you can do a lot more than you realized. So one of the things employers expect from someone with "work experience" is the elimination of the flake reflex—the ability to get things done, with no excuses.

The other thing you get from work experience is an understanding of what work is, and in particular, how intrinsically horrible it is. Fundamentally the equation is a brutal one: you have to spend most of your waking hours doing stuff someone else wants, or starve. There are a few places where the work is so interesting that this is concealed, because what other people want done happens to coincide with what you want to work on. But you only have to imagine what would happen if they diverged to see the underlying reality.

It's not so much that adults lie to kids about this as never explain it. They never explain what the deal is with money. You know from an early age that you'll have some sort of job, because everyone asks what you're going to "be" when you grow up. What they don't tell you is that as a kid you're sitting on the shoulders of someone else who's treading water, and that starting working means you get thrown into the water on your own, and have to start treading water yourself or sink. "Being" something is incidental; the immediate problem is not to drown.

The relationship between work and money tends to dawn on you only gradually. At least it did for me. One's first thought tends to be simply "This sucks. I'm in debt. Plus I have to get up on monday and go to work." Gradually you realize that these two things are as tightly connected as only a market can make them.

So the most important advantage 24 year old founders have over 20 year old founders is that they know what they're trying to avoid. To the average undergrad the idea of getting rich translates into buying Ferraris, or being admired. To someone who has learned from experience about the relationship between money and work, it translates to something way more important: it means you get to opt out of the brutal equation that governs the lives of 99.9% of people. Getting rich means you can stop treading water.

Someone who gets this will work much harder at making a startup succeed—with the proverbial energy of a drowning man, in fact. But understanding the

relationship between money and work also changes the way you work. You don't get money just for working, but for doing things other people want. Someone who's figured that out will automatically focus more on the user. And that cures the other half of the class-project syndrome. After you've been working for a while, you yourself tend to measure what you've done the same way the market does.

Of course, you don't have to spend years working to learn this stuff. If you're sufficiently perceptive you can grasp these things while you're still in school. Sam Altman did. He must have, because Loopt is no class project. And as his example suggests, this can be valuable knowledge. At a minimum, if you get this stuff, you already have most of what you gain from the "work experience" employers consider so desirable. But of course if you really get it, you can use this information in a way that's more valuable to you than that.

Now

So suppose you think you might start a startup at some point, either when you graduate or a few years after. What should you do now? For both jobs and grad school, there are ways to prepare while you're in college. If you want to get a job when you graduate, you should get summer jobs at places you'd like to work. If you want to go to grad school, it will help to work on research projects as an undergrad. What's the equivalent for startups? How do you keep your options maximally open?

One thing you can do while you're still in school is to learn how startups work. Unfortunately that's not easy. Few if any colleges have classes about startups. There may be business school classes on entrepreneurship, as they call it over there, but these are likely to be a waste of time. Business schools like to talk about startups, but philosophically they're at the opposite end of the spectrum. Most books on startups also seem to be useless. I've looked at a few and none get it right. Books in most fields are written by people who know the subject from experience, but for startups there's a unique problem: by definition the founders of successful startups don't need to write books to make money. As a result most books on the subject end up being written by people who don't understand it.

So I'd be skeptical of classes and books. The way to learn about startups is by watching them in action, preferably by working at one. How do you do that as an undergrad? Probably by sneaking in through the back door. Just hang around a lot and gradually start doing things for them. Most startups are (or should be) very cautious about hiring. Every hire increases the burn rate, and bad hires early on are hard to recover from. However, startups usually have a fairly informal atmosphere, and there's always a lot that needs to be done. If you just start doing stuff for them, many will be too busy to shoo you away. You can thus gradually work your way into their confidence, and maybe turn it into an official job later, or not, whichever you prefer. This won't work for all startups, but it would work for most I've known.

Number two, make the most of the great advantage of school: the wealth of co-

founders. Look at the people around you and ask yourself which you'd like to work with. When you apply that test, you may find you get surprising results. You may find you'd prefer the quiet guy you've mostly ignored to someone who seems impressive but has an attitude to match. I'm not suggesting you suck up to people you don't really like because you think one day they'll be successful. Exactly the opposite, in fact: you should only start a startup with someone you like, because a startup will put your friendship through a stress test. I'm just saying you should think about who you really admire and hang out with them, instead of whoever circumstances throw you together with.

Another thing you can do is learn skills that will be useful to you in a startup. These may be different from the skills you'd learn to get a job. For example, thinking about getting a job will make you want to learn programming languages you think employers want, like Java and C++. Whereas if you start a startup, you get to pick the language, so you have to think about which will actually let you get the most done. If you use that test you might end up learning Ruby or Python instead.

But the most important skill for a startup founder isn't a programming technique. It's a knack for understanding users and figuring out how to give them what they want. I know I repeat this, but that's because it's so important. And it's a skill you can learn, though perhaps habit might be a better word. Get into the habit of thinking of software as having users. What do those users want? What would make them say wow?

This is particularly valuable for undergrads, because the concept of users is missing from most college programming classes. The way you get taught programming in college would be like teaching writing as grammar, without mentioning that its purpose is to communicate something to an audience. Fortunately an audience for software is now only an http request away. So in addition to the programming you do for your classes, why not build some kind of website people will find useful? At the very least it will teach you how to write software with users. In the best case, it might not just be preparation for a startup, but the startup itself, like it was for Yahoo and Google.

Notes

[1] Even the desire to protect one's children seems weaker, judging from things people have historically done to their kids rather than risk their community's disapproval. (I assume we still do things that will be regarded in the future as barbaric, but historical abuses are easier for us to see.)

[2] Worrying that Y Combinator makes founders move for 3 months also suggests one underestimates how hard it is to start a startup. You're going to have to put up with much greater inconveniences than that.

[3] Most employee agreements say that any idea relating to the company's present or potential future business belongs to them. Often as not the second clause could include any possible startup, and anyone doing due diligence for an investor or acquirer will assume the worst.

To be safe either (a) don't use code written while you were still employed in your previous job, or (b) get your employer to renounce, in writing, any claim to the code you write for your side project. Many will consent to (b) rather than lose a prized employee. The downside is that you'll have to tell them exactly what your project does.

[4] Geshke and Warnock only founded Adobe because Xerox ignored them. If Xerox had used what they built, they would probably never have left PARC.

Thanks to Jessica Livingston and Robert Morris for reading drafts of this, and to Jeff Arnold and the SIPB for inviting me to speak.

 Comment on this essay.

Chinese Translation

Arabic Translation

The 18 Mistakes That Kill Startups

October 2006

In the Q & A period after a recent talk, someone asked what made startups fail. After standing there gaping for a few seconds I realized this was kind of a trick question. It's equivalent to asking how to make a startup succeed — if you avoid every cause of failure, you succeed — and that's too big a question to answer on the fly.

Afterwards I realized it could be helpful to look at the problem from this direction. If you have a list of all the things you shouldn't do, you can turn that into a recipe for succeeding just by negating. And this form of list may be more useful in practice. It's easier to catch yourself doing something you shouldn't than always to remember to do something you should. [1]

In a sense there's just one mistake that kills startups: not making something users want. If you make something users want, you'll probably be fine, whatever else you do or don't do. And if you don't make something users want, then you're dead, whatever else you do or don't do. So really this is a list of 18 things that cause startups not to make something users want. Nearly all failure funnels through that.

1. Single Founder

Have you ever noticed how few successful startups were founded by just one person? Even companies you think of as having one founder, like Oracle, usually turn out to have more. It seems unlikely this is a coincidence.

What's wrong with having one founder? To start with, it's a vote of no confidence. It probably means the founder couldn't talk any of his friends into starting the company with him. That's pretty alarming, because his friends are the ones who know him best.

But even if the founder's friends were all wrong and the company is a good bet, he's still at a disadvantage. Starting a startup is too hard for one person. Even if you could do all the work yourself, you need colleagues to brainstorm with, to talk you out of stupid decisions, and to cheer you up when things go wrong.

The last one might be the most important. The low points in a startup are so low

that few could bear them alone. When you have multiple founders, esprit de corps binds them together in a way that seems to violate conservation laws. Each thinks "I can't let my friends down." This is one of the most powerful forces in human nature, and it's missing when there's just one founder.

2. Bad Location

Startups prosper in some places and not others. Silicon Valley dominates, then Boston, then Seattle, Austin, Denver, and New York. After that there's not much. Even in New York the number of startups per capita is probably a 20th of what it is in Silicon Valley. In towns like Houston and Chicago and Detroit it's too small to measure.

Why is the falloff so sharp? Probably for the same reason it is in other industries. What's the sixth largest fashion center in the US? The sixth largest center for oil, or finance, or publishing? Whatever they are they're probably so far from the top that it would be misleading even to call them centers.

It's an interesting question why cities become startup hubs, but the reason startups prosper in them is probably the same as it is for any industry: that's where the experts are. Standards are higher; people are more sympathetic to what you're doing; the kind of people you want to hire want to live there; supporting industries are there; the people you run into in chance meetings are in the same business. Who knows exactly how these factors combine to boost startups in Silicon Valley and squish them in Detroit, but it's clear they do from the number of startups per capita in each.

3. Marginal Niche

Most of the groups that apply to Y Combinator suffer from a common problem: choosing a small, obscure niche in the hope of avoiding competition.

If you watch little kids playing sports, you notice that below a certain age they're afraid of the ball. When the ball comes near them their instinct is to avoid it. I didn't make a lot of catches as an eight year old outfielder, because whenever a fly ball came my way, I used to close my eyes and hold my glove up more for protection than in the hope of catching it.

Choosing a marginal project is the startup equivalent of my eight year old strategy for dealing with fly balls. If you make anything good, you're going to have competitors, so you may as well face that. You can only avoid competition by avoiding good ideas.

I think this shrinking from big problems is mostly unconscious. It's not that people think of grand ideas but decide to pursue smaller ones because they seem safer. Your unconscious won't even let you think of grand ideas. So the solution may be to think about ideas without involving yourself. What would be a great idea for *someone else* to do as a startup?

4. Derivative Idea

Many of the applications we get are imitations of some existing company. That's one source of ideas, but not the best. If you look at the origins of successful startups, few were started in imitation of some other startup. Where did they get their ideas? Usually from some specific, unsolved problem the founders identified.

Our startup made software for making online stores. When we started it, there wasn't any; the few sites you could order from were hand-made at great expense by web consultants. We knew that if online shopping ever took off, these sites would have to be generated by software, so we wrote some. Pretty straightforward.

It seems like the best problems to solve are ones that affect you personally. Apple happened because Steve Wozniak wanted a computer, Google because Larry and Sergey couldn't find stuff online, Hotmail because Sabeer Bhatia and Jack Smith couldn't exchange email at work.

So instead of copying the Facebook, with some variation that the Facebook rightly ignored, look for ideas from the other direction. Instead of starting from companies and working back to the problems they solved, look for problems and imagine the company that might solve them. [2] What do people complain about? What do you wish there was?

5. Obstinacy

In some fields the way to succeed is to have a vision of what you want to achieve, and to hold true to it no matter what setbacks you encounter. Starting startups is not one of them. The stick-to-your-vision approach works for something like winning an Olympic gold medal, where the problem is well-defined. Startups are more like science, where you need to follow the trail wherever it leads.

So don't get too attached to your original plan, because it's probably wrong. Most successful startups end up doing something different than they originally intended — often so different that it doesn't even seem like the same company. You have to be prepared to see the better idea when it arrives. And the hardest part of that is often discarding your old idea.

But openness to new ideas has to be tuned just right. Switching to a new idea every week will be equally fatal. Is there some kind of external test you can use? One is to ask whether the ideas represent some kind of progression. If in each new idea you're able to re-use most of what you built for the previous ones, then you're probably in a process that converges. Whereas if you keep restarting from scratch, that's a bad sign.

Fortunately there's someone you can ask for advice: your users. If you're thinking about turning in some new direction and your users seem excited about it, it's

probably a good bet.

6. Hiring Bad Programmers

I forgot to include this in the early versions of the list, because nearly all the founders I know are programmers. This is not a serious problem for them. They might accidentally hire someone bad, but it's not going to kill the company. In a pinch they can do whatever's required themselves.

But when I think about what killed most of the startups in the e-commerce business back in the 90s, it was bad programmers. A lot of those companies were started by business guys who thought the way startups worked was that you had some clever idea and then hired programmers to implement it. That's actually much harder than it sounds — almost impossibly hard in fact — because business guys can't tell which are the good programmers. They don't even get a shot at the best ones, because no one really good wants a job implementing the vision of a business guy.

In practice what happens is that the business guys choose people they think are good programmers (it says here on his resume that he's a Microsoft Certified Developer) but who aren't. Then they're mystified to find that their startup lumbers along like a World War II bomber while their competitors scream past like jet fighters. This kind of startup is in the same position as a big company, but without the advantages.

So how do you pick good programmers if you're not a programmer? I don't think there's an answer. I was about to say you'd have to find a good programmer to help you hire people. But if you can't recognize good programmers, how would you even do that?

7. Choosing the Wrong Platform

A related problem (since it tends to be done by bad programmers) is choosing the wrong platform. For example, I think a lot of startups during the Bubble killed themselves by deciding to build server-based applications on Windows. Hotmail was still running on FreeBSD for years after Microsoft bought it, presumably because Windows couldn't handle the load. If Hotmail's founders had chosen to use Windows, they would have been swamped.

PayPal only just dodged this bullet. After they merged with X.com, the new CEO wanted to switch to Windows — even after PayPal cofounder Max Levchin showed that their software scaled only 1% as well on Windows as Unix. Fortunately for PayPal they switched CEOs instead.

Platform is a vague word. It could mean an operating system, or a programming language, or a "framework" built on top of a programming language. It implies something that both supports and limits, like the foundation of a house.

The scary thing about platforms is that there are always some that seem to outsiders to be fine, responsible choices and yet, like Windows in the 90s, will destroy you if you choose them. Java applets were probably the most spectacular example. This was supposed to be the new way of delivering applications. Presumably it killed just about 100% of the startups who believed that.

How do you pick the right platforms? The usual way is to hire good programmers and let them choose. But there is a trick you could use if you're not a programmer: visit a top computer science department and see what they use in research projects.

8. Slowness in Launching

Companies of all sizes have a hard time getting software done. It's intrinsic to the medium; software is always 85% done. It takes an effort of will to push through this and get something released to users. [3]

Startups make all kinds of excuses for delaying their launch. Most are equivalent to the ones people use for procrastinating in everyday life. There's something that needs to happen first. Maybe. But if the software were 100% finished and ready to launch at the push of a button, would they still be waiting?

One reason to launch quickly is that it forces you to actually *finish* some quantum of work. Nothing is truly finished till it's released; you can see that from the rush of work that's always involved in releasing anything, no matter how finished you thought it was. The other reason you need to launch is that it's only by bouncing your idea off users that you fully understand it.

Several distinct problems manifest themselves as delays in launching: working too slowly; not truly understanding the problem; fear of having to deal with users; fear of being judged; working on too many different things; excessive perfectionism. Fortunately you can combat all of them by the simple expedient of forcing yourself to launch *something* fairly quickly.

9. Launching Too Early

Launching too slowly has probably killed a hundred times more startups than launching too fast, but it is possible to launch too fast. The danger here is that you ruin your reputation. You launch something, the early adopters try it out, and if it's no good they may never come back.

So what's the minimum you need to launch? We suggest startups think about what they plan to do, identify a core that's both (a) useful on its own and (b) something that can be incrementally expanded into the whole project, and then get that done as soon as possible.

This is the same approach I (and many other programmers) use for writing software. Think about the overall goal, then start by writing the smallest subset of

it that does anything useful. If it's a subset, you'll have to write it anyway, so in the worst case you won't be wasting your time. But more likely you'll find that implementing a working subset is both good for morale and helps you see more clearly what the rest should do.

The early adopters you need to impress are fairly tolerant. They don't expect a newly launched product to do everything; it just has to do *something*.

10. Having No Specific User in Mind

You can't build things users like without understanding them. I mentioned earlier that the most successful startups seem to have begun by trying to solve a problem their founders had. Perhaps there's a rule here: perhaps you create wealth in proportion to how well you understand the problem you're solving, and the problems you understand best are your own. [4]

That's just a theory. What's not a theory is the converse: if you're trying to solve problems you don't understand, you're hosed.

And yet a surprising number of founders seem willing to assume that someone, they're not sure exactly who, will want what they're building. Do the founders want it? No, they're not the target market. Who is? Teenagers. People interested in local events (that one is a perennial tarpit). Or "business" users. What business users? Gas stations? Movie studios? Defense contractors?

You can of course build something for users other than yourself. We did. But you should realize you're stepping into dangerous territory. You're flying on instruments, in effect, so you should (a) consciously shift gears, instead of assuming you can rely on your intuitions as you ordinarily would, and (b) look at the instruments.

In this case the instruments are the users. When designing for other people you have to be empirical. You can no longer guess what will work; you have to find users and measure their responses. So if you're going to make something for teenagers or "business" users or some other group that doesn't include you, you have to be able to talk some specific ones into using what you're making. If you can't, you're on the wrong track.

11. Raising Too Little Money

Most successful startups take funding at some point. Like having more than one founder, it seems a good bet statistically. How much should you take, though?

Startup funding is measured in time. Every startup that isn't profitable (meaning nearly all of them, initially) has a certain amount of time left before the money runs out and they have to stop. This is sometimes referred to as runway, as in "How much runway do you have left?" It's a good metaphor because it reminds you that when the money runs out you're going to be airborne or dead.

Too little money means not enough to get airborne. What airborne means depends on the situation. Usually you have to advance to a visibly higher level: if all you have is an idea, a working prototype; if you have a prototype, launching; if you're launched, significant growth. It depends on investors, because until you're profitable that's who you have to convince.

So if you take money from investors, you have to take enough to get to the next step, whatever that is. [5] Fortunately you have some control over both how much you spend and what the next step is. We advise startups to set both low, initially: spend practically nothing, and make your initial goal simply to build a solid prototype. This gives you maximum flexibility.

12. Spending Too Much

It's hard to distinguish spending too much from raising too little. If you run out of money, you could say either was the cause. The only way to decide which to call it is by comparison with other startups. If you raised five million and ran out of money, you probably spent too much.

Burning through too much money is not as common as it used to be. Founders seem to have learned that lesson. Plus it keeps getting cheaper to start a startup. So as of this writing few startups spend too much. None of the ones we've funded have. (And not just because we make small investments; many have gone on to raise further rounds.)

The classic way to burn through cash is by hiring a lot of people. This bites you twice: in addition to increasing your costs, it slows you down—so money that's getting consumed faster has to last longer. Most hackers understand why that happens; Fred Brooks explained it in The Mythical Man-Month.

We have three general suggestions about hiring: (a) don't do it if you can avoid it, (b) pay people with equity rather than salary, not just to save money, but because you want the kind of people who are committed enough to prefer that, and (c) only hire people who are either going to write code or go out and get users, because those are the only things you need at first.

13. Raising Too Much Money

It's obvious how too little money could kill you, but is there such a thing as having too much?

Yes and no. The problem is not so much the money itself as what comes with it. As one VC who spoke at Y Combinator said, "Once you take several million dollars of my money, the clock is ticking." If VCs fund you, they're not going to let you just put the money in the bank and keep operating as two guys living on ramen. They want that money to go to work. [6] At the very least you'll move into proper office space and hire more people. That will change the atmosphere, and not entirely for

the better. Now most of your people will be employees rather than founders. They won't be as committed; they'll need to be told what to do; they'll start to engage in office politics.

When you raise a lot of money, your company moves to the suburbs and has kids.

Perhaps more dangerously, once you take a lot of money it gets harder to change direction. Suppose your initial plan was to sell something to companies. After taking VC money you hire a sales force to do that. What happens now if you realize you should be making this for consumers instead of businesses? That's a completely different kind of selling. What happens, in practice, is that you don't realize that. The more people you have, the more you stay pointed in the same direction.

Another drawback of large investments is the time they take. The time required to raise money grows with the amount. [7] When the amount rises into the millions, investors get very cautious. VCs never quite say yes or no; they just engage you in an apparently endless conversation. Raising VC scale investments is thus a huge time sink — more work, probably, than the startup itself. And you don't want to be spending all your time talking to investors while your competitors are spending theirs building things.

We advise founders who go on to seek VC money to take the first reasonable deal they get. If you get an offer from a reputable firm at a reasonable valuation with no unusually onerous terms, just take it and get on with building the company. [8] Who cares if you could get a 30% better deal elsewhere? Economically, startups are an all-or-nothing game. Bargain-hunting among investors is a waste of time.

14. Poor Investor Management

As a founder, you have to manage your investors. You shouldn't ignore them, because they may have useful insights. But neither should you let them run the company. That's supposed to be your job. If investors had sufficient vision to run the companies they fund, why didn't they start them?

Pissing off investors by ignoring them is probably less dangerous than caving in to them. In our startup, we erred on the ignoring side. A lot of our energy got drained away in disputes with investors instead of going into the product. But this was less costly than giving in, which would probably have destroyed the company. If the founders know what they're doing, it's better to have half their attention focused on the product than the full attention of investors who don't.

How hard you have to work on managing investors usually depends on how much money you've taken. When you raise VC-scale money, the investors get a great deal of control. If they have a board majority, they're literally your bosses. In the more common case, where founders and investors are equally represented and the deciding vote is cast by neutral outside directors, all the investors have to do is convince the outside directors and they control the company.

If things go well, this shouldn't matter. So long as you seem to be advancing rapidly, most investors will leave you alone. But things don't always go smoothly in startups. Investors have made trouble even for the most successful companies. One of the most famous examples is Apple, whose board made a nearly fatal blunder in firing Steve Jobs. Apparently even Google got a lot of grief from their investors early on.

15. Sacrificing Users to (Supposed) Profit

When I said at the beginning that if you make something users want, you'll be fine, you may have noticed I didn't mention anything about having the right business model. That's not because making money is unimportant. I'm not suggesting that founders start companies with no chance of making money in the hope of unloading them before they tank. The reason we tell founders not to worry about the business model initially is that making something people want is so much harder.

I don't know why it's so hard to make something people want. It seems like it should be straightforward. But you can tell it must be hard by how few startups do it.

Because making something people want is so much harder than making money from it, you should leave business models for later, just as you'd leave some trivial but messy feature for version 2. In version 1, solve the core problem. And the core problem in a startup is how to create wealth (= how much people want something x the number who want it), not how to convert that wealth into money.

The companies that win are the ones that put users first. Google, for example. They made search work, then worried about how to make money from it. And yet some startup founders still think it's irresponsible not to focus on the business model from the beginning. They're often encouraged in this by investors whose experience comes from less malleable industries.

It *is* irresponsible not to think about business models. It's just ten times more irresponsible not to think about the product.

16. Not Wanting to Get Your Hands Dirty

Nearly all programmers would rather spend their time writing code and have someone else handle the messy business of extracting money from it. And not just the lazy ones. Larry and Sergey apparently felt this way too at first. After developing their new search algorithm, the first thing they tried was to get some other company to buy it.

Start a company? Yech. Most hackers would rather just have ideas. But as Larry and Sergey found, there's not much of a market for ideas. No one trusts an idea till you embody it in a product and use that to grow a user base. Then they'll pay

big time.

Maybe this will change, but I doubt it will change much. There's nothing like users for convincing acquirers. It's not just that the risk is decreased. The acquirers are human, and they have a hard time paying a bunch of young guys millions of dollars just for being clever. When the idea is embodied in a company with a lot of users, they can tell themselves they're buying the users rather than the cleverness, and this is easier for them to swallow. [9]

If you're going to attract users, you'll probably have to get up from your computer and go find some. It's unpleasant work, but if you can make yourself do it you have a much greater chance of succeeding. In the first batch of startups we funded, in the summer of 2005, most of the founders spent all their time building their applications. But there was one who was away half the time talking to executives at cell phone companies, trying to arrange deals. Can you imagine anything more painful for a hacker? [10] But it paid off, because this startup seems the most successful of that group by an order of magnitude.

If you want to start a startup, you have to face the fact that you can't just hack. At least one hacker will have to spend some of the time doing business stuff.

17. Fights Between Founders

Fights between founders are surprisingly common. About 20% of the startups we've funded have had a founder leave. It happens so often that we've reversed our attitude to vesting. We still don't require it, but now we advise founders to vest so there will be an orderly way for people to quit.

A founder leaving doesn't necessarily kill a startup, though. Plenty of successful startups have had that happen. [11] Fortunately it's usually the least committed founder who leaves. If there are three founders and one who was lukewarm leaves, big deal. If you have two and one leaves, or a guy with critical technical skills leaves, that's more of a problem. But even that is survivable. Blogger got down to one person, and they bounced back.

Most of the disputes I've seen between founders could have been avoided if they'd been more careful about who they started a company with. Most disputes are not due to the situation but the people. Which means they're inevitable. And most founders who've been burned by such disputes probably had misgivings, which they suppressed, when they started the company. Don't suppress misgivings. It's much easier to fix problems before the company is started than after. So don't include your housemate in your startup because he'd feel left out otherwise. Don't start a company with someone you dislike because they have some skill you need and you worry you won't find anyone else. The people are the most important ingredient in a startup, so don't compromise there.

18. A Half-Hearted Effort

The failed startups you hear most about are the spectacular flameouts. Those are actually the elite of failures. The most common type is not the one that makes spectacular mistakes, but the one that doesn't do much of anything — the one we never even hear about, because it was some project a couple guys started on the side while working on their day jobs, but which never got anywhere and was gradually abandoned.

Statistically, if you want to avoid failure, it would seem like the most important thing is to quit your day job. Most founders of failed startups don't quit their day jobs, and most founders of successful ones do. If startup failure were a disease, the CDC would be issuing bulletins warning people to avoid day jobs.

Does that mean you should quit your day job? Not necessarily. I'm guessing here, but I'd guess that many of these would-be founders may not have the kind of determination it takes to start a company, and that in the back of their minds, they know it. The reason they don't invest more time in their startup is that they know it's a bad investment. [12]

I'd also guess there's some band of people who could have succeeded if they'd taken the leap and done it full-time, but didn't. I have no idea how wide this band is, but if the winner/borderline/hopeless progression has the sort of distribution you'd expect, the number of people who could have made it, if they'd quit their day job, is probably an order of magnitude larger than the number who do make it. [13]

If that's true, most startups that could succeed fail because the founders don't devote their whole efforts to them. That certainly accords with what I see out in the world. Most startups fail because they don't make something people want, and the reason most don't is that they don't try hard enough.

In other words, starting startups is just like everything else. The biggest mistake you can make is not to try hard enough. To the extent there's a secret to success, it's not to be in denial about that.

Notes

[1] This is not a complete list of the causes of failure, just those you can control. There are also several you can't, notably ineptitude and bad luck.

[2] Ironically, one variant of the Facebook that might work is a facebook exclusively for college students.

[3] Steve Jobs tried to motivate people by saying "Real artists ship." This is a fine sentence, but unfortunately not true. Many famous works of art are unfinished. It's true in fields that have hard deadlines, like architecture and filmmaking, but even there people tend to be tweaking stuff till it's yanked out of their hands.

[4] There's probably also a second factor: startup founders tend to be at the leading edge of technology, so problems they face are probably especially valuable.

[5] You should take more than you think you'll need, maybe 50% to 100% more, because software takes longer to write and deals longer to close than you expect.

[6] Since people sometimes call us VCs, I should add that we're not. VCs invest large amounts of other people's money. We invest small amounts of our own, like angel investors.

[7] Not linearly of course, or it would take forever to raise five million dollars. In practice it just feels like it takes forever.

Though if you include the cases where VCs don't invest, it would literally take forever in the median case. And maybe we should, because the danger of chasing large investments is not just that they take a long time. That's the *best* case. The real danger is that you'll expend a lot of time and get nothing.

[8] Some VCs will offer you an artificially low valuation to see if you have the balls to ask for more. It's lame that VCs play such games, but some do. If you're dealing with one of those you should push back on the valuation a bit.

[9] Suppose YouTube's founders had gone to Google in 2005 and told them "Google Video is badly designed. Give us $10 million and we'll tell you all the mistakes you made." They would have gotten the royal raspberry. Eighteen months later Google paid $1.6 billion for the same lesson, partly because they could then tell themselves that they were buying a phenomenon, or a community, or some vague thing like that.

I don't mean to be hard on Google. They did better than their competitors, who may have now missed the video boat entirely.

[10] Yes, actually: dealing with the government. But phone companies are up there.

[11] Many more than most people realize, because companies don't advertise this. Did you know Apple originally had three founders?

[12] I'm not dissing these people. I don't have the determination myself. I've twice come close to starting startups since Viaweb, and both times I bailed because I realized that without the spur of poverty I just wasn't willing to endure the stress of a startup.

[13] So how do you know whether you're in the category of people who should quit their day job, or the presumably larger one who shouldn't? I got to the point of saying that this was hard to judge for yourself and that you should seek outside advice, before realizing that that's what we do. We think of ourselves as investors,

but viewed from the other direction Y Combinator is a service for advising people whether or not to quit their day job. We could be mistaken, and no doubt often are, but we do at least bet money on our conclusions.

Thanks to Sam Altman, Jessica Livingston, Greg McAdoo, and Robert Morris for reading drafts of this.

Japanese Translation

Spanish Translation

Romanian Translation

Chinese Translation

Arabic Translation

How Art Can Be Good

December 2006

I grew up believing that taste is just a matter of personal preference. Each person has things they like, but no one's preferences are any better than anyone else's. There is no such thing as *good* taste.

Like a lot of things I grew up believing, this turns out to be false, and I'm going to try to explain why.

One problem with saying there's no such thing as good taste is that it also means there's no such thing as good art. If there were good art, then people who liked it would have better taste than people who didn't. So if you discard taste, you also have to discard the idea of art being good, and artists being good at making it.

It was pulling on that thread that unravelled my childhood faith in relativism. When you're trying to make things, taste becomes a practical matter. You have to decide what to do next. Would it make the painting better if I changed that part? If there's no such thing as better, it doesn't matter what you do. In fact, it doesn't matter if you paint at all. You could just go out and buy a ready-made blank canvas. If there's no such thing as good, that would be just as great an achievement as the ceiling of the Sistine Chapel. Less laborious, certainly, but if you can achieve the same level of performance with less effort, surely that's more impressive, not less.

Yet that doesn't seem quite right, does it?

Audience

I think the key to this puzzle is to remember that art has an audience. Art has a purpose, which is to interest its audience. Good art (like good anything) is art that achieves its purpose particularly well. The meaning of "interest" can vary. Some

works of art are meant to shock, and others to please; some are meant to jump out at you, and others to sit quietly in the background. But all art has to work on an audience, and—here's the critical point—members of the audience share things in common.

For example, nearly all humans find human faces engaging. It seems to be wired into us. Babies can recognize faces practically from birth. In fact, faces seem to have co-evolved with our interest in them; the face is the body's billboard. So all other things being equal, a painting with faces in it will interest people more than one without. [1]

One reason it's easy to believe that taste is merely personal preference is that, if it isn't, how do you pick out the people with better taste? There are billions of people, each with their own opinion; on what grounds can you prefer one to another? [2]

But if audiences have a lot in common, you're not in a position of having to choose one out of a random set of individual biases, because the set isn't random. All humans find faces engaging—practically by definition: face recognition is in our DNA. And so having a notion of good art, in the sense of art that does its job well, doesn't require you to pick out a few individuals and label their opinions as correct. No matter who you pick, they'll find faces engaging.

Of course, space aliens probably wouldn't find human faces engaging. But there might be other things they shared in common with us. The most likely source of examples is math. I expect space aliens would agree with us most of the time about which of two proofs was better. Erdos thought so. He called a maximally elegant proof one out of God's book, and presumably God's book is universal. [3]

Once you start talking about audiences, you don't have to argue simply that there are or aren't standards of taste. Instead tastes are a series of concentric rings, like ripples in a pond. There are some things that will appeal to you and your friends, others that will appeal to most people your age, others that will appeal to most humans, and perhaps others that would appeal to most sentient beings (whatever that means).

The picture is slightly more complicated than that, because in the middle of the pond there are overlapping sets of ripples. For example, there might be things that appealed particularly to men, or to people from a certain culture.

If good art is art that interests its audience, then when you talk about art being good, you also have to say for what audience. So is it meaningless to talk about art simply being good or bad? No, because one audience is the set of all possible humans. I think that's the audience people are implicitly talking about when they say a work of art is good: they mean it would engage any human. [4]

And that is a meaningful test, because although, like any everyday concept, "human" is fuzzy around the edges, there are a lot of things practically all humans

have in common. In addition to our interest in faces, there's something special about primary colors for nearly all of us, because it's an artifact of the way our eyes work. Most humans will also find images of 3D objects engaging, because that also seems to be built into our visual perception. [5] And beneath that there's edge-finding, which makes images with definite shapes more engaging than mere blur.

Humans have a lot more in common than this, of course. My goal is not to compile a complete list, just to show that there's some solid ground here. People's preferences aren't random. So an artist working on a painting and trying to decide whether to change some part of it doesn't have to think "Why bother? I might as well flip a coin." Instead he can ask "What would make the painting more interesting to people?" And the reason you can't equal Michelangelo by going out and buying a blank canvas is that the ceiling of the Sistine Chapel is more interesting to people.

A lot of philosophers have had a hard time believing it was possible for there to be objective standards for art. It seemed obvious that beauty, for example, was something that happened in the head of the observer, not something that was a property of objects. It was thus "subjective" rather than "objective." But in fact if you narrow the definition of beauty to something that works a certain way on humans, and you observe how much humans have in common, it turns out to be a property of objects after all. You don't have to choose between something being a property of the subject or the object if subjects all react similarly. Being good art is thus a property of objects as much as, say, being toxic to humans is: it's good art if it consistently affects humans in a certain way.

Error

So could we figure out what the best art is by taking a vote? After all, if appealing to humans is the test, we should be able to just ask them, right?

Well, not quite. For products of nature that might work. I'd be willing to eat the apple the world's population had voted most delicious, and I'd probably be willing to visit the beach they voted most beautiful, but having to look at the painting they voted the best would be a crapshoot.

Man-made stuff is different. For one thing, artists, unlike apple trees, often deliberately try to trick us. Some tricks are quite subtle. For example, any work of art sets expectations by its level of finish. You don't expect photographic accuracy in something that looks like a quick sketch. So one widely used trick, especially among illustrators, is to intentionally make a painting or drawing look like it was done faster than it was. The average person looks at it and thinks: how amazingly skillful. It's like saying something clever in a conversation as if you'd thought of it on the spur of the moment, when in fact you'd worked it out the day before.

Another much less subtle influence is brand. If you go to see the Mona Lisa, you'll probably be disappointed, because it's hidden behind a thick glass wall and

surrounded by a frenzied crowd taking pictures of themselves in front of it. At best you can see it the way you see a friend across the room at a crowded party. The Louvre might as well replace it with copy; no one would be able to tell. And yet the Mona Lisa is a small, dark painting. If you found people who'd never seen an image of it and sent them to a museum in which it was hanging among other paintings with a tag labelling it as a portrait by an unknown fifteenth century artist, most would walk by without giving it a second look.

For the average person, brand dominates all other factors in the judgement of art. Seeing a painting they recognize from reproductions is so overwhelming that their response to it as a painting is drowned out.

And then of course there are the tricks people play on themselves. Most adults looking at art worry that if they don't like what they're supposed to, they'll be thought uncultured. This doesn't just affect what they claim to like; they actually make themselves like things they're supposed to.

That's why you can't just take a vote. Though appeal to people is a meaningful test, in practice you can't measure it, just as you can't find north using a compass with a magnet sitting next to it. There are sources of error so powerful that if you take a vote, all you're measuring is the error.

We can, however, approach our goal from another direction, by using ourselves as guinea pigs. You're human. If you want to know what the basic human reaction to a piece of art would be, you can at least approach that by getting rid of the sources of error in your own judgements.

For example, while anyone's reaction to a famous painting will be warped at first by its fame, there are ways to decrease its effects. One is to come back to the painting over and over. After a few days the fame wears off, and you can start to see it as a painting. Another is to stand close. A painting familiar from reproductions looks more familiar from ten feet away; close in you see details that get lost in reproductions, and which you're therefore seeing for the first time.

There are two main kinds of error that get in the way of seeing a work of art: biases you bring from your own circumstances, and tricks played by the artist. Tricks are straightforward to correct for. Merely being aware of them usually prevents them from working. For example, when I was ten I used to be very impressed by airbrushed lettering that looked like shiny metal. But once you study how it's done, you see that it's a pretty cheesy trick—one of the sort that relies on pushing a few visual buttons really hard to temporarily overwhelm the viewer. It's like trying to convince someone by shouting at them.

The way not to be vulnerable to tricks is to explicitly seek out and catalog them. When you notice a whiff of dishonesty coming from some kind of art, stop and figure out what's going on. When someone is obviously pandering to an audience that's easily fooled, whether it's someone making shiny stuff to impress ten year olds, or someone making conspicuously avant-garde stuff to impress would-be

intellectuals, learn how they do it. Once you've seen enough examples of specific types of tricks, you start to become a connoisseur of trickery in general, just as professional magicians are.

What counts as a trick? Roughly, it's something done with contempt for the audience. For example, the guys designing Ferraris in the 1950s were probably designing cars that they themselves admired. Whereas I suspect over at General Motors the marketing people are telling the designers, "Most people who buy SUVs do it to seem manly, not to drive off-road. So don't worry about the suspension; just make that sucker as big and tough-looking as you can." [6]

I think with some effort you can make yourself nearly immune to tricks. It's harder to escape the influence of your own circumstances, but you can at least move in that direction. The way to do it is to travel widely, in both time and space. If you go and see all the different kinds of things people like in other cultures, and learn about all the different things people have liked in the past, you'll probably find it changes what you like. I doubt you could ever make yourself into a completely universal person, if only because you can only travel in one direction in time. But if you find a work of art that would appeal equally to your friends, to people in Nepal, and to the ancient Greeks, you're probably onto something.

My main point here is not how to have good taste, but that there can even be such a thing. And I think I've shown that. There is such a thing as good art. It's art that interests its human audience, and since humans have a lot in common, what interests them is not random. Since there's such a thing as good art, there's also such a thing as good taste, which is the ability to recognize it.

If we were talking about the taste of apples, I'd agree that taste is just personal preference. Some people like certain kinds of apples and others like other kinds, but how can you say that one is right and the other wrong? [7]

The thing is, art isn't apples. Art is man-made. It comes with a lot of cultural baggage, and in addition the people who make it often try to trick us. Most people's judgement of art is dominated by these extraneous factors; they're like someone trying to judge the taste of apples in a dish made of equal parts apples and jalapeno peppers. All they're tasting is the peppers. So it turns out you can pick out some people and say that they have better taste than others: they're the ones who actually taste art like apples.

Or to put it more prosaically, they're the people who (a) are hard to trick, and (b) don't just like whatever they grew up with. If you could find people who'd eliminated all such influences on their judgement, you'd probably still see variation in what they liked. But because humans have so much in common, you'd also find they agreed on a lot. They'd nearly all prefer the ceiling of the Sistine Chapel to a blank canvas.

Making It

I wrote this essay because I was tired of hearing "taste is subjective" and wanted to kill it once and for all. Anyone who makes things knows intuitively that's not true. When you're trying to make art, the temptation to be lazy is as great as in any other kind of work. Of course it matters to do a good job. And yet you can see how great a hold "taste is subjective" has even in the art world by how nervous it makes people to talk about art being good or bad. Those whose jobs require them to judge art, like curators, mostly resort to euphemisms like "significant" or "important" or (getting dangerously close) "realized." [8]

I don't have any illusions that being able to talk about art being good or bad will cause the people who talk about it to have anything more useful to say. Indeed, one of the reasons "taste is subjective" found such a receptive audience is that, historically, the things people have said about good taste have generally been such nonsense.

It's not for the people who talk about art that I want to free the idea of good art, but for those who make it. Right now, ambitious kids going to art school run smack into a brick wall. They arrive hoping one day to be as good as the famous artists they've seen in books, and the first thing they learn is that the concept of good has been retired. Instead everyone is just supposed to explore their own personal vision. [9]

When I was in art school, we were looking one day at a slide of some great fifteenth century painting, and one of the students asked "Why don't artists paint like that now?" The room suddenly got quiet. Though rarely asked out loud, this question lurks uncomfortably in the back of every art student's mind. It was as if someone had brought up the topic of lung cancer in a meeting within Philip Morris.

"Well," the professor replied, "we're interested in different questions now." He was a pretty nice guy, but at the time I couldn't help wishing I could send him back to fifteenth century Florence to explain in person to Leonardo & Co. how we had moved beyond their early, limited concept of art. Just imagine that conversation.

In fact, one of the reasons artists in fifteenth century Florence made such great things was that they believed you could make great things. [10] They were intensely competitive and were always trying to outdo one another, like mathematicians or physicists today—maybe like anyone who has ever done anything really well.

The idea that you could make great things was not just a useful illusion. They were actually right. So the most important consequence of realizing there can be good art is that it frees artists to try to make it. To the ambitious kids arriving at art school this year hoping one day to make great things, I say: don't believe it when they tell you this is a naive and outdated ambition. There is such a thing as good art, and if you try to make it, there are people who will notice.

Notes

[1] This is not to say, of course, that good paintings must have faces in them, just that everyone's visual piano has that key on it. There are situations in which you want to avoid faces, precisely because they attract so much attention. But you can see how universally faces work by their prevalence in advertising.

[2] The other reason it's easy to believe is that it makes people feel good. To a kid, this idea is crack. In every other respect they're constantly being told that they have a lot to learn. But in this they're perfect. Their opinion carries the same weight as any adult's. You should probably question anything you believed as a kid that you'd want to believe this much.

[3] It's conceivable that the elegance of proofs is quantifiable, in the sense that there may be some formal measure that turns out to coincide with mathematicians' judgements. Perhaps it would be worth trying to make a formal language for proofs in which those considered more elegant consistently came out shorter (perhaps after being macroexpanded or compiled).

[4] Maybe it would be possible to make art that would appeal to space aliens, but I'm not going to get into that because (a) it's too hard to answer, and (b) I'm satisfied if I can establish that good art is a meaningful idea for human audiences.

[5] If early abstract paintings seem more interesting than later ones, it may be because the first abstract painters were trained to paint from life, and their hands thus tended to make the kind of gestures you use in representing physical things. In effect they were saying "scaramara" instead of "uebfgbsb."

[6] It's a bit more complicated, because sometimes artists unconsciously use tricks by imitating art that does.

[7] I phrased this in terms of the taste of apples because if people can see the apples, they can be fooled. When I was a kid most apples were a variety called Red Delicious that had been bred to look appealing in stores, but which didn't taste very good.

[8] To be fair, curators are in a difficult position. If they're dealing with recent art, they have to include things in shows that they think are bad. That's because the test for what gets included in shows is basically the market price, and for recent art that is largely determined by successful businessmen and their wives. So it's not always intellectual dishonesty that makes curators and dealers use neutral-sounding language.

[9] What happens in practice is that everyone gets really good at *talking* about art. As the art itself gets more random, the effort that would have gone into the work goes instead into the intellectual sounding theory behind it. "My work represents

an exploration of gender and sexuality in an urban context," etc. Different people win at that game.

[10] There were several other reasons, including that Florence was then the richest and most sophisticated city in the world, and that they lived in a time before photography had (a) killed portraiture as a source of income and (b) made brand the dominant factor in the sale of art.

Incidentally, I'm not saying that good art = fifteenth century European art. I'm not saying we should make what they made, but that we should work like they worked. There are fields now in which many people work with the same energy and honesty that fifteenth century artists did, but art is not one of them.

Thanks to Trevor Blackwell, Jessica Livingston, and Robert Morris for reading drafts of this, and to Paul Watson for permission to use the image at the top.

Japanese Translation

Simplified Chinese Translation

Learning from Founders

January 2007

(Foreword to Jessica Livingston's Founders at Work.)

Apparently sprinters reach their highest speed right out of the blocks, and spend the rest of the race slowing down. The winners slow down the least. It's that way with most startups too. The earliest phase is usually the most productive. That's when they have the really big ideas. Imagine what Apple was like when 100% of its employees were either Steve Jobs or Steve Wozniak.

The striking thing about this phase is that it's completely different from most people's idea of what business is like. If you looked in people's heads (or stock photo collections) for images representing "business," you'd get images of people dressed up in suits, groups sitting around conference tables looking serious, Powerpoint presentations, people producing thick reports for one another to read. Early stage startups are the exact opposite of this. And yet they're probably the most productive part of the whole economy.

Why the disconnect? I think there's a general principle at work here: the less energy people expend on performance, the more they expend on appearances to compensate. More often than not the energy they expend on seeming impressive makes their actual performance worse. A few years ago I read an article in which a car magazine modified the "sports" model of some production car to get the fastest possible standing quarter mile. You know how they did it? They cut off all the crap the manufacturer had bolted onto the car to make it *look* fast.

Business is broken the same way that car was. The effort that goes into looking productive is not merely wasted, but actually makes organizations less productive. Suits, for example. Suits do not help people to think better. I bet most executives at big companies do their best thinking when they wake up on Sunday morning and go downstairs in their bathrobe to make a cup of coffee. That's when you have ideas. Just imagine what a company would be like if people could think that well at work. People do in startups, at least some of the time. (Half the time you're in a panic because your servers are on fire, but the other half you're thinking as deeply as most people only get to sitting alone on a Sunday morning.)

Ditto for most of the other differences between startups and what passes for productivity in big companies. And yet conventional ideas of professionalism have

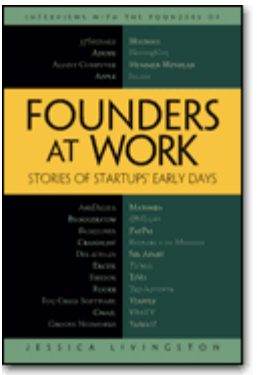
such an iron grip on our minds that even startup founders are affected by them. In our startup, when outsiders came to visit we tried hard to seem "professional." We'd clean up our offices, wear better clothes, try to arrange that a lot of people were there during conventional office hours. In fact, programming didn't get done by well-dressed people at clean desks during office hours. It got done by badly dressed people (I was notorious for programmming wearing just a towel) in offices strewn with junk at 2 in the morning. But no visitor would understand that. Not even investors, who are supposed to be able to recognize real productivity when they see it. Even we were affected by the conventional wisdom. We thought of ourselves as impostors, succeeding despite being totally unprofessional. It was as if we'd created a Formula 1 car but felt sheepish because it didn't look like a car was supposed to look.

In the car world, there are at least some people who know that a high performance car looks like a Formula 1 racecar, not a sedan with giant rims and a fake spoiler bolted to the trunk. Why not in business? Probably because startups are so small. The really dramatic growth happens when a startup only has three or four people, so only three or four people see that, whereas tens of thousands see business as it's practiced by Boeing or Philip Morris.

This book can help fix that problem, by showing everyone what, till now, only a handful people got to see: what happens in the first year of a startup. This is what real productivity looks like. This is the Formula 1 racecar. It looks weird, but it goes fast.

Of course, big companies won't be able to do everything these startups do. In big companies there's always going to be more politics, and less scope for individual decisions. But seeing what startups are really like will at least show other organizations what to aim for. The time may soon be coming when instead of startups trying to seem more corporate, corporations will try to seem more like startups. That would be a good thing.

Japanese Translation



Founders at Work

There can't be more than a couple thousand people who know first-hand what happens in the first month of a successful startup. Jessica Livingston got them to tell us. So despite the interview format, this is really a how-to book. It is probably the single most valuable book a startup founder could read.

Is It Worth Being Wise?

February 2007

A few days ago I finally figured out something I've wondered about for 25 years: the relationship between wisdom and intelligence. Anyone can see they're not the same by the number of people who are smart, but not very wise. And yet intelligence and wisdom do seem related. How?

What is wisdom? I'd say it's knowing what to do in a lot of situations. I'm not trying to make a deep point here about the true nature of wisdom, just to figure out how we use the word. A wise person is someone who usually knows the right thing to do.

And yet isn't being smart also knowing what to do in certain situations? For example, knowing what to do when the teacher tells your elementary school class to add all the numbers from 1 to 100? [1]

Some say wisdom and intelligence apply to different types of problems—wisdom to human problems and intelligence to abstract ones. But that isn't true. Some wisdom has nothing to do with people: for example, the wisdom of the engineer who knows certain structures are less prone to failure than others. And certainly smart people can find clever solutions to human problems as well as abstract ones. [2]

Another popular explanation is that wisdom comes from experience while intelligence is innate. But people are not simply wise in proportion to how much experience they have. Other things must contribute to wisdom besides experience, and some may be innate: a reflective disposition, for example.

Neither of the conventional explanations of the difference between wisdom and intelligence stands up to scrutiny. So what is the difference? If we look at how people use the words "wise" and "smart," what they seem to mean is different shapes of performance.

Curve

"Wise" and "smart" are both ways of saying someone knows what to do. The difference is that "wise" means one has a high average outcome across all situations, and "smart" means one does spectacularly well in a few. That is, if you

had a graph in which the x axis represented situations and the y axis the outcome, the graph of the wise person would be high overall, and the graph of the smart person would have high peaks.

The distinction is similar to the rule that one should judge talent at its best and character at its worst. Except you judge intelligence at its best, and wisdom by its average. That's how the two are related: they're the two different senses in which the same curve can be high.

So a wise person knows what to do in most situations, while a smart person knows what to do in situations where few others could. We need to add one more qualification: we should ignore cases where someone knows what to do because they have inside information. [3] But aside from that, I don't think we can get much more specific without starting to be mistaken.

Nor do we need to. Simple as it is, this explanation predicts, or at least accords with, both of the conventional stories about the distinction between wisdom and intelligence. Human problems are the most common type, so being good at solving those is key in achieving a high average outcome. And it seems natural that a high average outcome depends mostly on experience, but that dramatic peaks can only be achieved by people with certain rare, innate qualities; nearly anyone can learn to be a good swimmer, but to be an Olympic swimmer you need a certain body type.

This explanation also suggests why wisdom is such an elusive concept: there's no such thing. "Wise" means something—that one is on average good at making the right choice. But giving the name "wisdom" to the supposed quality that enables one to do that doesn't mean such a thing exists. To the extent "wisdom" means anything, it refers to a grab-bag of qualities as various as self-discipline, experience, and empathy. [4]

Likewise, though "intelligent" means something, we're asking for trouble if we insist on looking for a single thing called "intelligence." And whatever its components, they're not all innate. We use the word "intelligent" as an indication of ability: a smart person can grasp things few others could. It does seem likely there's some inborn predisposition to intelligence (and wisdom too), but this predisposition is not itself intelligence.

One reason we tend to think of intelligence as inborn is that people trying to measure it have concentrated on the aspects of it that are most measurable. A quality that's inborn will obviously be more convenient to work with than one that's influenced by experience, and thus might vary in the course of a study. The problem comes when we drag the word "intelligence" over onto what they're measuring. If they're measuring something inborn, they can't be measuring intelligence. Three year olds aren't smart. When we describe one as smart, it's shorthand for "smarter than other three year olds."

Split

Perhaps it's a technicality to point out that a predisposition to intelligence is not the same as intelligence. But it's an important technicality, because it reminds us that we can become smarter, just as we can become wiser.

The alarming thing is that we may have to choose between the two.

If wisdom and intelligence are the average and peaks of the same curve, then they converge as the number of points on the curve decreases. If there's just one point, they're identical: the average and maximum are the same. But as the number of points increases, wisdom and intelligence diverge. And historically the number of points on the curve seems to have been increasing: our ability is tested in an ever wider range of situations.

In the time of Confucius and Socrates, people seem to have regarded wisdom, learning, and intelligence as more closely related than we do. Distinguishing between "wise" and "smart" is a modern habit. [5] And the reason we do is that they've been diverging. As knowledge gets more specialized, there are more points on the curve, and the distinction between the spikes and the average becomes sharper, like a digital image rendered with more pixels.

One consequence is that some old recipes may have become obsolete. At the very least we have to go back and figure out if they were really recipes for wisdom or intelligence. But the really striking change, as intelligence and wisdom drift apart, is that we may have to decide which we prefer. We may not be able to optimize for both simultaneously.

Society seems to have voted for intelligence. We no longer admire the sage—not the way people did two thousand years ago. Now we admire the genius. Because in fact the distinction we began with has a rather brutal converse: just as you can be smart without being very wise, you can be wise without being very smart. That doesn't sound especially admirable. That gets you James Bond, who knows what to do in a lot of situations, but has to rely on Q for the ones involving math.

Intelligence and wisdom are obviously not mutually exclusive. In fact, a high average may help support high peaks. But there are reasons to believe that at some point you have to choose between them. One is the example of very smart people, who are so often unwise that in popular culture this now seems to be regarded as the rule rather than the exception. Perhaps the absent-minded professor is wise in his way, or wiser than he seems, but he's not wise in the way Confucius or Socrates wanted people to be. [6]

New

For both Confucius and Socrates, wisdom, virtue, and happiness were necessarily related. The wise man was someone who knew what the right choice was and always made it; to be the right choice, it had to be morally right; he was therefore always happy, knowing he'd done the best he could. I can't think of many ancient

philosophers who would have disagreed with that, so far as it goes.

"The superior man is always happy; the small man sad," said Confucius. [7]

Whereas a few years ago I read an interview with a mathematician who said that most nights he went to bed discontented, feeling he hadn't made enough progress. [8] The Chinese and Greek words we translate as "happy" didn't mean exactly what we do by it, but there's enough overlap that this remark contradicts them.

Is the mathematician a small man because he's discontented? No; he's just doing a kind of work that wasn't very common in Confucius's day.

Human knowledge seems to grow fractally. Time after time, something that seemed a small and uninteresting area—experimental error, even—turns out, when examined up close, to have as much in it as all knowledge up to that point. Several of the fractal buds that have exploded since ancient times involve inventing and discovering new things. Math, for example, used to be something a handful of people did part-time. Now it's the career of thousands. And in work that involves making new things, some old rules don't apply.

Recently I've spent some time advising people, and there I find the ancient rule still works: try to understand the situation as well as you can, give the best advice you can based on your experience, and then don't worry about it, knowing you did all you could. But I don't have anything like this serenity when I'm writing an essay. Then I'm worried. What if I run out of ideas? And when I'm writing, four nights out of five I go to bed discontented, feeling I didn't get enough done.

Advising people and writing are fundamentally different types of work. When people come to you with a problem and you have to figure out the right thing to do, you don't (usually) have to invent anything. You just weigh the alternatives and try to judge which is the prudent choice. But *prudence* can't tell me what sentence to write next. The search space is too big.

Someone like a judge or a military officer can in much of his work be guided by duty, but duty is no guide in making things. Makers depend on something more precarious: inspiration. And like most people who lead a precarious existence, they tend to be worried, not contented. In that respect they're more like the small man of Confucius's day, always one bad harvest (or ruler) away from starvation. Except instead of being at the mercy of weather and officials, they're at the mercy of their own imagination.

Limits

To me it was a relief just to realize it might be ok to be discontented. The idea that a successful person should be happy has thousands of years of momentum behind it. If I was any good, why didn't I have the easy confidence winners are supposed to have? But that, I now believe, is like a runner asking "If I'm such a good athlete, why do I feel so tired?" Good runners still get tired; they just get tired at

higher speeds.

People whose work is to invent or discover things are in the same position as the runner. There's no way for them to do the best they can, because there's no limit to what they could do. The closest you can come is to compare yourself to other people. But the better you do, the less this matters. An undergrad who gets something published feels like a star. But for someone at the top of the field, what's the test of doing well? Runners can at least compare themselves to others doing exactly the same thing; if you win an Olympic gold medal, you can be fairly content, even if you think you could have run a bit faster. But what is a novelist to do?

Whereas if you're doing the kind of work in which problems are presented to you and you have to choose between several alternatives, there's an upper bound on your performance: choosing the best every time. In ancient societies, nearly all work seems to have been of this type. The peasant had to decide whether a garment was worth mending, and the king whether or not to invade his neighbor, but neither was expected to invent anything. In principle they could have; the king could have invented firearms, then invaded his neighbor. But in practice innovations were so rare that they weren't expected of you, any more than goalkeepers are expected to score goals. [9] In practice, it seemed as if there was a correct decision in every situation, and if you made it you'd done your job perfectly, just as a goalkeeper who prevents the other team from scoring is considered to have played a perfect game.

In this world, wisdom seemed paramount. [10] Even now, most people do work in which problems are put before them and they have to choose the best alternative. But as knowledge has grown more specialized, there are more and more types of work in which people have to make up new things, and in which performance is therefore unbounded. Intelligence has become increasingly important relative to wisdom because there is more room for spikes.

Recipes

Another sign we may have to choose between intelligence and wisdom is how different their recipes are. Wisdom seems to come largely from curing childish qualities, and intelligence largely from cultivating them.

Recipes for wisdom, particularly ancient ones, tend to have a remedial character. To achieve wisdom one must cut away all the debris that fills one's head on emergence from childhood, leaving only the important stuff. Both self-control and experience have this effect: to eliminate the random biases that come from your own nature and from the circumstances of your upbringing respectively. That's not all wisdom is, but it's a large part of it. Much of what's in the sage's head is also in the head of every twelve year old. The difference is that in the head of the twelve year old it's mixed together with a lot of random junk.

The path to intelligence seems to be through working on hard problems. You

develop intelligence as you might develop muscles, through exercise. But there can't be too much compulsion here. No amount of discipline can replace genuine curiosity. So cultivating intelligence seems to be a matter of identifying some bias in one's character—some tendency to be interested in certain types of things—and nurturing it. Instead of obliterating your idiosyncrasies in an effort to make yourself a neutral vessel for the truth, you select one and try to grow it from a seedling into a tree.

The wise are all much alike in their wisdom, but very smart people tend to be smart in distinctive ways.

Most of our educational traditions aim at wisdom. So perhaps one reason schools work badly is that they're trying to make intelligence using recipes for wisdom. Most recipes for wisdom have an element of subjection. At the very least, you're supposed to do what the teacher says. The more extreme recipes aim to break down your individuality the way basic training does. But that's not the route to intelligence. Whereas wisdom comes through humility, it may actually help, in cultivating intelligence, to have a mistakenly high opinion of your abilities, because that encourages you to keep working. Ideally till you realize how mistaken you were.

(The reason it's hard to learn new skills late in life is not just that one's brain is less malleable. Another probably even worse obstacle is that one has higher standards.)

I realize we're on dangerous ground here. I'm not proposing the primary goal of education should be to increase students' "self-esteem." That just breeds laziness. And in any case, it doesn't really fool the kids, not the smart ones. They can tell at a young age that a contest where everyone wins is a fraud.

A teacher has to walk a narrow path: you want to encourage kids to come up with things on their own, but you can't simply applaud everything they produce. You have to be a good audience: appreciative, but not too easily impressed. And that's a lot of work. You have to have a good enough grasp of kids' capacities at different ages to know when to be surprised.

That's the opposite of traditional recipes for education. Traditionally the student is the audience, not the teacher; the student's job is not to invent, but to absorb some prescribed body of material. (The use of the term "recitation" for sections in some colleges is a fossil of this.) The problem with these old traditions is that they're too much influenced by recipes for wisdom.

Different

I deliberately gave this essay a provocative title; of course it's worth being wise. But I think it's important to understand the relationship between intelligence and wisdom, and particularly what seems to be the growing gap between them. That way we can avoid applying rules and standards to intelligence that are really

meant for wisdom. These two senses of "knowing what to do" are more different than most people realize. The path to wisdom is through discipline, and the path to intelligence through carefully selected self-indulgence. Wisdom is universal, and intelligence idiosyncratic. And while wisdom yields calmness, intelligence much of the time leads to discontentment.

That's particularly worth remembering. A physicist friend recently told me half his department was on Prozac. Perhaps if we acknowledge that some amount of frustration is inevitable in certain kinds of work, we can mitigate its effects. Perhaps we can box it up and put it away some of the time, instead of letting it flow together with everyday sadness to produce what seems an alarmingly large pool. At the very least, we can avoid being discontented about being discontented.

If you feel exhausted, it's not necessarily because there's something wrong with you. Maybe you're just running fast.

Notes

[1] Gauss was supposedly asked this when he was 10. Instead of laboriously adding together the numbers like the other students, he saw that they consisted of 50 pairs that each summed to 101 (100 + 1, 99 + 2, etc), and that he could just multiply 101 by 50 to get the answer, 5050.

[2] A variant is that intelligence is the ability to solve problems, and wisdom the judgement to know how to use those solutions. But while this is certainly an important relationship between wisdom and intelligence, it's not the *distinction between* them. Wisdom is useful in solving problems too, and intelligence can help in deciding what to do with the solutions.

[3] In judging both intelligence and wisdom we have to factor out some knowledge. People who know the combination of a safe will be better at opening it than people who don't, but no one would say that was a test of intelligence or wisdom.

But knowledge overlaps with wisdom and probably also intelligence. A knowledge of human nature is certainly part of wisdom. So where do we draw the line?

Perhaps the solution is to discount knowledge that at some point has a sharp drop in utility. For example, understanding French will help you in a large number of situations, but its value drops sharply as soon as no one else involved knows French. Whereas the value of understanding vanity would decline more gradually.

The knowledge whose utility drops sharply is the kind that has little relation to other knowledge. This includes mere conventions, like languages and safe

combinations, and also what we'd call "random" facts, like movie stars' birthdays, or how to distinguish 1956 from 1957 Studebakers.

[4] People seeking some single thing called "wisdom" have been fooled by grammar. Wisdom is just knowing the right thing to do, and there are a hundred and one different qualities that help in that. Some, like selflessness, might come from meditating in an empty room, and others, like a knowledge of human nature, might come from going to drunken parties.

Perhaps realizing this will help dispel the cloud of semi-sacred mystery that surrounds wisdom in so many people's eyes. The mystery comes mostly from looking for something that doesn't exist. And the reason there have historically been so many different schools of thought about how to achieve wisdom is that they've focused on different components of it.

When I use the word "wisdom" in this essay, I mean no more than whatever collection of qualities helps people make the right choice in a wide variety of situations.

[5] Even in English, our sense of the word "intelligence" is surprisingly recent. Predecessors like "understanding" seem to have had a broader meaning.

[6] There is of course some uncertainty about how closely the remarks attributed to Confucius and Socrates resemble their actual opinions. I'm using these names as we use the name "Homer," to mean the hypothetical people who said the things attributed to them.

[7] *Analects* VII:36, Fung trans.

Some translators use "calm" instead of "happy." One source of difficulty here is that present-day English speakers have a different idea of happiness from many older societies. Every language probably has a word meaning "how one feels when things are going well," but different cultures react differently when things go well. We react like children, with smiles and laughter. But in a more reserved society, or in one where life was tougher, the reaction might be a quiet contentment.

[8] It may have been Andrew Wiles, but I'm not sure. If anyone remembers such an interview, I'd appreciate hearing from you.

[9] Confucius claimed proudly that he had never invented anything—that he had simply passed on an accurate account of ancient traditions. [*Analects* VII:1] It's hard for us now to appreciate how important a duty it must have been in preliterate societies to remember and pass on the group's accumulated knowledge. Even in Confucius's time it still seems to have been the first duty of the scholar.

[10] The bias toward wisdom in ancient philosophy may be exaggerated by the fact that, in both Greece and China, many of the first philosophers (including Confucius and Plato) saw themselves as teachers of administrators, and so thought

disproportionately about such matters. The few people who did invent things, like storytellers, must have seemed an outlying data point that could be ignored.

Thanks to Trevor Blackwell, Sarah Harlin, Jessica Livingston, and Robert Morris for reading drafts of this.

Polish Translation

French Translation

Russian Translation

Russian Translation

Why to Not Not Start a Startup

March 2007

(This essay is derived from talks at the 2007 Startup School and the Berkeley CSUA.)

We've now been doing Y Combinator long enough to have some data about success rates. Our first batch, in the summer of 2005, had eight startups in it. Of those eight, it now looks as if at least four succeeded. Three have been acquired: Reddit was a merger of two, Reddit and Infogami, and a third was acquired that we can't talk about yet. Another from that batch was Loopt, which is doing so well they could probably be acquired in about ten minutes if they wanted to.

So about half the founders from that first summer, less than two years ago, are now rich, at least by their standards. (One thing you learn when you get rich is that there are many degrees of it.)

I'm not ready to predict our success rate will stay as high as 50%. That first batch could have been an anomaly. But we should be able to do better than the oft-quoted (and probably made up) standard figure of 10%. I'd feel safe aiming at 25%.

Even the founders who fail don't seem to have such a bad time. Of those first eight startups, three are now probably dead. In two cases the founders just went on to do other things at the end of the summer. I don't think they were traumatized by the experience. The closest to a traumatic failure was Kiko, whose founders kept working on their startup for a whole year before being squashed by Google Calendar. But they ended up happy. They sold their software on eBay for a quarter of a million dollars. After they paid back their angel investors, they had about a year's salary each. [1] Then they immediately went on to start a new and much more exciting startup, Justin.TV.

So here is an even more striking statistic: 0% of that first batch had a terrible experience. They had ups and downs, like every startup, but I don't think any would have traded it for a job in a cubicle. And that statistic is probably not an anomaly. Whatever our long-term success rate ends up being, I think the rate of people who wish they'd gotten a regular job will stay close to 0%.

The big mystery to me is: why don't more people start startups? If nearly everyone who does it prefers it to a regular job, and a significant percentage get rich, why doesn't everyone want to do this? A lot of people think we get thousands of applications for each funding cycle. In fact we usually only get several hundred. Why don't more people apply? And while it must seem to anyone watching this world that startups are popping up like crazy, the number is small compared to the number of people with the necessary skills. The great majority of programmers still go straight from college to cubicle, and stay there.

It seems like people are not acting in their own interest. What's going on? Well, I can answer that. Because of Y Combinator's position at the very start of the venture funding process, we're probably the world's leading experts on the psychology of people who aren't sure if they want to start a company.

There's nothing wrong with being unsure. If you're a hacker thinking about starting a startup and hesitating before taking the leap, you're part of a grand tradition. Larry and Sergey seem to have felt the same before they started Google, and so did Jerry and Filo before they started Yahoo. In fact, I'd guess the most successful startups are the ones started by uncertain hackers rather than gung-ho business guys.

We have some evidence to support this. Several of the most successful startups we've funded told us later that they only decided to apply at the last moment. Some decided only hours before the deadline.

The way to deal with uncertainty is to analyze it into components. Most people who are reluctant to do something have about eight different reasons mixed together in their heads, and don't know themselves which are biggest. Some will be justified and some bogus, but unless you know the relative proportion of each, you don't know whether your overall uncertainty is mostly justified or mostly bogus.

So I'm going to list all the components of people's reluctance to start startups, and explain which are real. Then would-be founders can use this as a checklist to examine their own feelings.

I admit my goal is to increase your self-confidence. But there are two things different here from the usual confidence-building exercise. One is that I'm motivated to be honest. Most people in the confidence-building business have already achieved their goal when you buy the book or pay to attend the seminar where they tell you how great you are. Whereas if I encourage people to start startups who shouldn't, I make my own life worse. If I encourage too many people to apply to Y Combinator, it just means more work for me, because I have to read all the applications.

The other thing that's going to be different is my approach. Instead of being positive, I'm going to be negative. Instead of telling you "come on, you can do it" I'm going to consider all the reasons you aren't doing it, and show why most (but not all) should be ignored. We'll start with the one everyone's born with.

1. Too young

A lot of people think they're too young to start a startup. Many are right. The median age worldwide is about 27, so probably a third of the population can truthfully say they're too young.

What's too young? One of our goals with Y Combinator was to discover the lower bound on the age of startup founders. It always seemed to us that investors were too conservative here—that they wanted to fund professors, when really they should be funding grad students or even undergrads.

The main thing we've discovered from pushing the edge of this envelope is not where the edge is, but how fuzzy it is. The outer limit may be as low as 16. We don't look beyond 18 because people younger than that can't legally enter into contracts. But the most successful founder we've funded so far, Sam Altman, was 19 at the time.

Sam Altman, however, is an outlying data point. When he was 19, he seemed like he had a 40 year old inside him. There are other 19 year olds who are 12 inside.

There's a reason we have a distinct word "adult" for people over a certain age. There is a threshold you cross. It's conventionally fixed at 21, but different people cross it at greatly varying ages. You're old enough to start a startup if you've crossed this threshold, whatever your age.

How do you tell? There are a couple tests adults use. I realized these tests existed after meeting Sam Altman, actually. I noticed that I felt like I was talking to someone much older. Afterward I wondered, what am I even measuring? What made him seem older?

One test adults use is whether you still have the kid flake reflex. When you're a little kid and you're asked to do something hard, you can cry and say "I can't do it" and the adults will probably let you off. As a kid there's a magic button you can press by saying "I'm just a kid" that will get you out of most difficult situations. Whereas adults, by definition, are not allowed to flake. They still do, of course, but when they do they're ruthlessly pruned.

The other way to tell an adult is by how they react to a challenge. Someone who's not yet an adult will tend to respond to a challenge from an adult in a way that acknowledges their dominance. If an adult says "that's a stupid idea," a kid will either crawl away with his tail between his legs, or rebel. But rebelling presumes inferiority as much as submission. The adult response to "that's a stupid idea," is simply to look the other person in the eye and say "Really? Why do you think so?"

There are a lot of adults who still react childishly to challenges, of course. What you don't often find are kids who react to challenges like adults. When you do, you've found an adult, whatever their age.

2. Too inexperienced

I once wrote that startup founders should be at least 23, and that people should work for another company for a few years before starting their own. I no longer believe that, and what changed my mind is the example of the startups we've funded.

I still think 23 is a better age than 21. But the best way to get experience if you're 21 is to start a startup. So, paradoxically, if you're too inexperienced to start a startup, what you should do is start one. That's a way more efficient cure for inexperience than a normal job. In fact, getting a normal job may actually make you less able to start a startup, by turning you into a tame animal who thinks he needs an office to work in and a product manager to tell him what software to write.

What really convinced me of this was the Kikos. They started a startup right out of college. Their inexperience caused them to make a lot of mistakes. But by the time we funded their second startup, a year later, they had become extremely formidable. They were certainly not tame animals. And there is no way they'd have grown so much if they'd spent that year working at Microsoft, or even Google. They'd still have been diffident junior programmers.

So now I'd advise people to go ahead and start startups right out of college. There's no better time to take risks than when you're young. Sure, you'll probably fail. But even failure will get you to the ultimate goal faster than getting a job.

It worries me a bit to be saying this, because in effect we're advising people to educate themselves by failing at our expense, but it's the truth.

3. Not determined enough

You need a lot of determination to succeed as a startup founder. It's probably the single best predictor of success.

Some people may not be determined enough to make it. It's hard for me to say for sure, because I'm so determined that I can't imagine what's going on in the heads of people who aren't. But I know they exist.

Most hackers probably underestimate their determination. I've seen a lot become visibly more determined as they get used to running a startup. I can think of several we've funded who would have been delighted at first to be bought for $2 million, but are now set on world domination.

How can you tell if you're determined enough, when Larry and Sergey themselves were unsure at first about starting a company? I'm guessing here, but I'd say the test is whether you're sufficiently driven to work on your own projects. Though they may have been unsure whether they wanted to start a company, it doesn't

seem as if Larry and Sergey were meek little research assistants, obediently doing their advisors' bidding. They started projects of their own.

4. Not smart enough

You may need to be moderately smart to succeed as a startup founder. But if you're worried about this, you're probably mistaken. If you're smart enough to worry that you might not be smart enough to start a startup, you probably are.

And in any case, starting a startup just doesn't require that much intelligence. Some startups do. You have to be good at math to write Mathematica. But most companies do more mundane stuff where the decisive factor is effort, not brains. Silicon Valley can warp your perspective on this, because there's a cult of smartness here. People who aren't smart at least try to act that way. But if you think it takes a lot of intelligence to get rich, try spending a couple days in some of the fancier bits of New York or LA.

If you don't think you're smart enough to start a startup doing something technically difficult, just write enterprise software. Enterprise software companies aren't technology companies, they're sales companies, and sales depends mostly on effort.

5. Know nothing about business

This is another variable whose coefficient should be zero. You don't need to know anything about business to start a startup. The initial focus should be the product. All you need to know in this phase is how to build things people want. If you succeed, you'll have to think about how to make money from it. But this is so easy you can pick it up on the fly.

I get a fair amount of flak for telling founders just to make something great and not worry too much about making money. And yet all the empirical evidence points that way: pretty much 100% of startups that make something popular manage to make money from it. And acquirers tell me privately that revenue is not what they buy startups for, but their strategic value. Which means, because they made something people want. Acquirers know the rule holds for them too: if users love you, you can always make money from that somehow, and if they don't, the cleverest business model in the world won't save you.

So why do so many people argue with me? I think one reason is that they hate the idea that a bunch of twenty year olds could get rich from building something cool that doesn't make any money. They just don't want that to be possible. But how possible it is doesn't depend on how much they want it to be.

For a while it annoyed me to hear myself described as some kind of irresponsible pied piper, leading impressionable young hackers down the road to ruin. But now I realize this kind of controversy is a sign of a good idea.

The most valuable truths are the ones most people don't believe. They're like undervalued stocks. If you start with them, you'll have the whole field to yourself. So when you find an idea you know is good but most people disagree with, you should not merely ignore their objections, but push aggressively in that direction. In this case, that means you should seek out ideas that would be popular but seem hard to make money from.

We'll bet a seed round you can't make something popular that we can't figure out how to make money from.

6. No cofounder

Not having a cofounder is a real problem. A startup is too much for one person to bear. And though we differ from other investors on a lot of questions, we all agree on this. All investors, without exception, are more likely to fund you with a cofounder than without.

We've funded two single founders, but in both cases we suggested their first priority should be to find a cofounder. Both did. But we'd have preferred them to have cofounders before they applied. It's not super hard to get a cofounder for a project that's just been funded, and we'd rather have cofounders committed enough to sign up for something super hard.

If you don't have a cofounder, what should you do? Get one. It's more important than anything else. If there's no one where you live who wants to start a startup with you, move where there are people who do. If no one wants to work with you on your current idea, switch to an idea people want to work on.

If you're still in school, you're surrounded by potential cofounders. A few years out it gets harder to find them. Not only do you have a smaller pool to draw from, but most already have jobs, and perhaps even families to support. So if you had friends in college you used to scheme about startups with, stay in touch with them as well as you can. That may help keep the dream alive.

It's possible you could meet a cofounder through something like a user's group or a conference. But I wouldn't be too optimistic. You need to work with someone to know whether you want them as a cofounder. [2]

The real lesson to draw from this is not how to find a cofounder, but that you should start startups when you're young and there are lots of them around.

7. No idea

In a sense, it's not a problem if you don't have a good idea, because most startups change their idea anyway. In the average Y Combinator startup, I'd guess 70% of the idea is new at the end of the first three months. Sometimes it's 100%.

In fact, we're so sure the founders are more important than the initial idea that

we're going to try something new this funding cycle. We're going to let people apply with no idea at all. If you want, you can answer the question on the application form that asks what you're going to do with "We have no idea." If you seem really good we'll accept you anyway. We're confident we can sit down with you and cook up some promising project.

Really this just codifies what we do already. We put little weight on the idea. We ask mainly out of politeness. The kind of question on the application form that we really care about is the one where we ask what cool things you've made. If what you've made is version one of a promising startup, so much the better, but the main thing we care about is whether you're good at making things. Being lead developer of a popular open source project counts almost as much.

That solves the problem if you get funded by Y Combinator. What about in the general case? Because in another sense, it is a problem if you don't have an idea. If you start a startup with no idea, what do you do next?

So here's the brief recipe for getting startup ideas. Find something that's missing in your own life, and supply that need—no matter how specific to you it seems. Steve Wozniak built himself a computer; who knew so many other people would want them? A need that's narrow but genuine is a better starting point than one that's broad but hypothetical. So even if the problem is simply that you don't have a date on Saturday night, if you can think of a way to fix that by writing software, you're onto something, because a lot of other people have the same problem.

8. No room for more startups

A lot of people look at the ever-increasing number of startups and think "this can't continue." Implicit in their thinking is a fallacy: that there is some limit on the number of startups there could be. But this is false. No one claims there's any limit on the number of people who can work for salary at 1000-person companies. Why should there be any limit on the number who can work for equity at 5-person companies? [3]

Nearly everyone who works is satisfying some kind of need. Breaking up companies into smaller units doesn't make those needs go away. Existing needs would probably get satisfied more efficiently by a network of startups than by a few giant, hierarchical organizations, but I don't think that would mean less opportunity, because satisfying current needs would lead to more. Certainly this tends to be the case in individuals. Nor is there anything wrong with that. We take for granted things that medieval kings would have considered effeminate luxuries, like whole buildings heated to spring temperatures year round. And if things go well, our descendants will take for granted things we would consider shockingly luxurious. There is no absolute standard for material wealth. Health care is a component of it, and that alone is a black hole. For the foreseeable future, people will want ever more material wealth, so there is no limit to the amount of work available for companies, and for startups in particular.

Usually the limited-room fallacy is not expressed directly. Usually it's implicit in statements like "there are only so many startups Google, Microsoft, and Yahoo can buy." Maybe, though the list of acquirers is a lot longer than that. And whatever you think of other acquirers, Google is not stupid. The reason big companies buy startups is that they've created something valuable. And why should there be any limit to the number of valuable startups companies can acquire, any more than there is a limit to the amount of wealth individual people want? Maybe there would be practical limits on the number of startups any one acquirer could assimilate, but if there is value to be had, in the form of upside that founders are willing to forgo in return for an immediate payment, acquirers will evolve to consume it. Markets are pretty smart that way.

9. Family to support

This one is real. I wouldn't advise anyone with a family to start a startup. I'm not saying it's a bad idea, just that I don't want to take responsibility for advising it. I'm willing to take responsibility for telling 22 year olds to start startups. So what if they fail? They'll learn a lot, and that job at Microsoft will still be waiting for them if they need it. But I'm not prepared to cross moms.

What you can do, if you have a family and want to start a startup, is start a consulting business you can then gradually turn into a product business. Empirically the chances of pulling that off seem very small. You're never going to produce Google this way. But at least you'll never be without an income.

Another way to decrease the risk is to join an existing startup instead of starting your own. Being one of the first employees of a startup is a lot like being a founder, in both the good ways and the bad. You'll be roughly 1/n^2 founder, where n is your employee number.

As with the question of cofounders, the real lesson here is to start startups when you're young.

10. Independently wealthy

This is my excuse for not starting a startup. Startups are stressful. Why do it if you don't need the money? For every "serial entrepreneur," there are probably twenty sane ones who think "Start another company? Are you crazy?"

I've come close to starting new startups a couple times, but I always pull back because I don't want four years of my life to be consumed by random schleps. I know this business well enough to know you can't do it half-heartedly. What makes a good startup founder so dangerous is his willingness to endure infinite schleps.

There is a bit of a problem with retirement, though. Like a lot of people, I like to work. And one of the many weird little problems you discover when you get rich is that a lot of the interesting people you'd like to work with are not rich. They need to work at something that pays the bills. Which means if you want to have them as

colleagues, you have to work at something that pays the bills too, even though you don't need to. I think this is what drives a lot of serial entrepreneurs, actually.

That's why I love working on Y Combinator so much. It's an excuse to work on something interesting with people I like.

11. Not ready for commitment

This was my reason for not starting a startup for most of my twenties. Like a lot of people that age, I valued freedom most of all. I was reluctant to do anything that required a commitment of more than a few months. Nor would I have wanted to do anything that completely took over my life the way a startup does. And that's fine. If you want to spend your time travelling around, or playing in a band, or whatever, that's a perfectly legitimate reason not to start a company.

If you start a startup that succeeds, it's going to consume at least three or four years. (If it fails, you'll be done a lot quicker.) So you shouldn't do it if you're not ready for commitments on that scale. Be aware, though, that if you get a regular job, you'll probably end up working there for as long as a startup would take, and you'll find you have much less spare time than you might expect. So if you're ready to clip on that ID badge and go to that orientation session, you may also be ready to start that startup.

12. Need for structure

I'm told there are people who need structure in their lives. This seems to be a nice way of saying they need someone to tell them what to do. I believe such people exist. There's plenty of empirical evidence: armies, religious cults, and so on. They may even be the majority.

If you're one of these people, you probably shouldn't start a startup. In fact, you probably shouldn't even go to work for one. In a good startup, you don't get told what to do very much. There may be one person whose job title is CEO, but till the company has about twelve people no one should be telling anyone what to do. That's too inefficient. Each person should just do what they need to without anyone telling them.

If that sounds like a recipe for chaos, think about a soccer team. Eleven people manage to work together in quite complicated ways, and yet only in occasional emergencies does anyone tell anyone else what to do. A reporter once asked David Beckham if there were any language problems at Real Madrid, since the players were from about eight different countries. He said it was never an issue, because everyone was so good they never had to talk. They all just did the right thing.

How do you tell if you're independent-minded enough to start a startup? If you'd bristle at the suggestion that you aren't, then you probably are.

13. Fear of uncertainty

Perhaps some people are deterred from starting startups because they don't like the uncertainty. If you go to work for Microsoft, you can predict fairly accurately what the next few years will be like—all too accurately, in fact. If you start a startup, anything might happen.

Well, if you're troubled by uncertainty, I can solve that problem for you: if you start a startup, it will probably fail. Seriously, though, this is not a bad way to think about the whole experience. Hope for the best, but expect the worst. In the worst case, it will at least be interesting. In the best case you might get rich.

No one will blame you if the startup tanks, so long as you made a serious effort. There may once have been a time when employers would regard that as a mark against you, but they wouldn't now. I asked managers at big companies, and they all said they'd prefer to hire someone who'd tried to start a startup and failed over someone who'd spent the same time working at a big company.

Nor will investors hold it against you, as long as you didn't fail out of laziness or incurable stupidity. I'm told there's a lot of stigma attached to failing in other places—in Europe, for example. Not here. In America, companies, like practically everything else, are disposable.

14. Don't realize what you're avoiding

One reason people who've been out in the world for a year or two make better founders than people straight from college is that they know what they're avoiding. If their startup fails, they'll have to get a job, and they know how much jobs suck.

If you've had summer jobs in college, you may think you know what jobs are like, but you probably don't. Summer jobs at technology companies are not real jobs. If you get a summer job as a waiter, that's a real job. Then you have to carry your weight. But software companies don't hire students for the summer as a source of cheap labor. They do it in the hope of recruiting them when they graduate. So while they're happy if you produce, they don't expect you to.

That will change if you get a real job after you graduate. Then you'll have to earn your keep. And since most of what big companies do is boring, you're going to have to work on boring stuff. Easy, compared to college, but boring. At first it may seem cool to get paid for doing easy stuff, after paying to do hard stuff in college. But that wears off after a few months. Eventually it gets demoralizing to work on dumb stuff, even if it's easy and you get paid a lot.

And that's not the worst of it. The thing that really sucks about having a regular job is the expectation that you're supposed to be there at certain times. Even Google is afflicted with this, apparently. And what this means, as everyone who's had a regular job can tell you, is that there are going to be times when you have absolutely no desire to work on anything, and you're going to have to go to work anyway and sit in front of your screen and pretend to. To someone who likes work,

as most good hackers do, this is torture.

In a startup, you skip all that. There's no concept of office hours in most startups. Work and life just get mixed together. But the good thing about that is that no one minds if you have a life at work. In a startup you can do whatever you want most of the time. If you're a founder, what you want to do most of the time is work. But you never have to pretend to.

If you took a nap in your office in a big company, it would seem unprofessional. But if you're starting a startup and you fall asleep in the middle of the day, your cofounders will just assume you were tired.

15. Parents want you to be a doctor

A significant number of would-be startup founders are probably dissuaded from doing it by their parents. I'm not going to say you shouldn't listen to them. Families are entitled to their own traditions, and who am I to argue with them? But I will give you a couple reasons why a safe career might not be what your parents really want for you.

One is that parents tend to be more conservative for their kids than they would be for themselves. This is actually a rational response to their situation. Parents end up sharing more of their kids' ill fortune than good fortune. Most parents don't mind this; it's part of the job; but it does tend to make them excessively conservative. And erring on the side of conservatism is still erring. In almost everything, reward is proportionate to risk. So by protecting their kids from risk, parents are, without realizing it, also protecting them from rewards. If they saw that, they'd want you to take more risks.

The other reason parents may be mistaken is that, like generals, they're always fighting the last war. If they want you to be a doctor, odds are it's not just because they want you to help the sick, but also because it's a prestigious and lucrative career. [4] But not so lucrative or prestigious as it was when their opinions were formed. When I was a kid in the seventies, a doctor was *the* thing to be. There was a sort of golden triangle involving doctors, Mercedes 450SLs, and tennis. All three vertices now seem pretty dated.

The parents who want you to be a doctor may simply not realize how much things have changed. Would they be that unhappy if you were Steve Jobs instead? So I think the way to deal with your parents' opinions about what you should do is to treat them like feature requests. Even if your only goal is to please them, the way to do that is not simply to give them what they ask for. Instead think about why they're asking for something, and see if there's a better way to give them what they need.

16. A job is the default

This leads us to the last and probably most powerful reason people get regular

jobs: it's the default thing to do. Defaults are enormously powerful, precisely because they operate without any conscious choice.

To almost everyone except criminals, it seems an axiom that if you need money, you should get a job. Actually this tradition is not much more than a hundred years old. Before that, the default way to make a living was by farming. It's a bad plan to treat something only a hundred years old as an axiom. By historical standards, that's something that's changing pretty rapidly.

We may be seeing another such change right now. I've read a lot of economic history, and I understand the startup world pretty well, and it now seems to me fairly likely that we're seeing the beginning of a change like the one from farming to manufacturing.

And you know what? If you'd been around when that change began (around 1000 in Europe) it would have seemed to nearly everyone that running off to the city to make your fortune was a crazy thing to do. Though serfs were in principle forbidden to leave their manors, it can't have been that hard to run away to a city. There were no guards patrolling the perimeter of the village. What prevented most serfs from leaving was that it seemed insanely risky. Leave one's plot of land? Leave the people you'd spent your whole life with, to live in a giant city of three or four thousand complete strangers? How would you live? How would you get food, if you didn't grow it?

Frightening as it seemed to them, it's now the default with us to live by our wits. So if it seems risky to you to start a startup, think how risky it once seemed to your ancestors to live as we do now. Oddly enough, the people who know this best are the very ones trying to get you to stick to the old model. How can Larry and Sergey say you should come work as their employee, when they didn't get jobs themselves?

Now we look back on medieval peasants and wonder how they stood it. How grim it must have been to till the same fields your whole life with no hope of anything better, under the thumb of lords and priests you had to give all your surplus to and acknowledge as your masters. I wouldn't be surprised if one day people look back on what we consider a normal job in the same way. How grim it would be to commute every day to a cubicle in some soulless office complex, and be told what to do by someone you had to acknowledge as a boss—someone who could call you into their office and say "take a seat," and you'd sit! Imagine having to ask *permission* to release software to users. Imagine being sad on Sunday afternoons because the weekend was almost over, and tomorrow you'd have to get up and go to work. How did they stand it?

It's exciting to think we may be on the cusp of another shift like the one from farming to manufacturing. That's why I care about startups. Startups aren't interesting just because they're a way to make a lot of money. I couldn't care less about other ways to do that, like speculating in securities. At most those are interesting the way puzzles are. There's more going on with startups. They may

represent one of those rare, historic shifts in the way wealth is created.

That's ultimately what drives us to work on Y Combinator. We want to make money, if only so we don't have to stop doing it, but that's not the main goal. There have only been a handful of these great economic shifts in human history. It would be an amazing hack to make one happen faster.

Notes

[1] The only people who lost were us. The angels had convertible debt, so they had first claim on the proceeds of the auction. Y Combinator only got 38 cents on the dollar.

[2] The best kind of organization for that might be an open source project, but those don't involve a lot of face to face meetings. Maybe it would be worth starting one that did.

[3] There need to be some number of big companies to acquire the startups, so the number of big companies couldn't decrease to zero.

[4] Thought experiment: If doctors did the same work, but as impoverished outcasts, which parents would still want their kids to be doctors?

Thanks to Trevor Blackwell, Jessica Livingston, and Robert Morris for reading drafts of this, to the founders of Zenter for letting me use their web-based PowerPoint killer even though it isn't launched yet, and to Ming-Hay Luk of the Berkeley CSUA for inviting me to speak.

 Comment on this essay.

Russian Translation

Japanese Translation

Korean Translation

Microsoft is Dead

April 2007

A few days ago I suddenly realized Microsoft was dead. I was talking to a young startup founder about how Google was different from Yahoo. I said that Yahoo had been warped from the start by their fear of Microsoft. That was why they'd positioned themselves as a "media company" instead of a technology company. Then I looked at his face and realized he didn't understand. It was as if I'd told him how much girls liked Barry Manilow in the mid 80s. Barry who?

Microsoft? He didn't say anything, but I could tell he didn't quite believe anyone would be frightened of them.

Microsoft cast a shadow over the software world for almost 20 years starting in the late 80s. I can remember when it was IBM before them. I mostly ignored this shadow. I never used Microsoft software, so it only affected me indirectly—for example, in the spam I got from botnets. And because I wasn't paying attention, I didn't notice when the shadow disappeared.

But it's gone now. I can sense that. No one is even afraid of Microsoft anymore. They still make a lot of money—so does IBM, for that matter. But they're not dangerous.

When did Microsoft die, and of what? I know they seemed dangerous as late as 2001, because I wrote an essay then about how they were less dangerous than they seemed. I'd guess they were dead by 2005. I know when we started Y Combinator we didn't worry about Microsoft as competition for the startups we funded. In fact, we've never even invited them to the demo days we organize for startups to present to investors. We invite Yahoo and Google and some other Internet companies, but we've never bothered to invite Microsoft. Nor has anyone there ever even sent us an email. They're in a different world.

What killed them? Four things, I think, all of them occurring simultaneously in the mid 2000s.

The most obvious is Google. There can only be one big man in town, and they're clearly it. Google is the most dangerous company now by far, in both the good and bad senses of the word. Microsoft can at best limp along afterward.

When did Google take the lead? There will be a tendency to push it back to their IPO in August 2004, but they weren't setting the terms of the debate then. I'd say they took the lead in 2005. Gmail was one of the things that put them over the edge. Gmail showed they could do more than search.

Gmail also showed how much you could do with web-based software, if you took advantage of what later came to be called "Ajax." And that was the second cause of Microsoft's death: everyone can see the desktop is over. It now seems inevitable that applications will live on the web—not just email, but everything, right up to Photoshop. Even Microsoft sees that now.

Ironically, Microsoft unintentionally helped create Ajax. The x in Ajax is from the XMLHttpRequest object, which lets the browser communicate with the server in the background while displaying a page. (Originally the only way to communicate with the server was to ask for a new page.) XMLHttpRequest was created by Microsoft in the late 90s because they needed it for Outlook. What they didn't realize was that it would be useful to a lot of other people too—in fact, to anyone who wanted to make web apps work like desktop ones.

The other critical component of Ajax is Javascript, the programming language that runs in the browser. Microsoft saw the danger of Javascript and tried to keep it broken for as long as they could. [1] But eventually the open source world won, by producing Javascript libraries that grew over the brokenness of Explorer the way a tree grows over barbed wire.

The third cause of Microsoft's death was broadband Internet. Anyone who cares can have fast Internet access now. And the bigger the pipe to the server, the less you need the desktop.

The last nail in the coffin came, of all places, from Apple. Thanks to OS X, Apple has come back from the dead in a way that is extremely rare in technology. [2] Their victory is so complete that I'm now surprised when I come across a computer running Windows. Nearly all the people we fund at Y Combinator use Apple laptops. It was the same in the audience at startup school. All the computer people use Macs or Linux now. Windows is for grandmas, like Macs used to be in the 90s. So not only does the desktop no longer matter, no one who cares about computers uses Microsoft's anyway.

And of course Apple has Microsoft on the run in music too, with TV and phones on the way.

I'm glad Microsoft is dead. They were like Nero or Commodus—evil in the way only inherited power can make you. Because remember, the Microsoft monopoly didn't begin with Microsoft. They got it from IBM. The software business was overhung by a monopoly from about the mid-1950s to about 2005. For practically its whole existence, that is. One of the reasons "Web 2.0" has such an air of euphoria about it is the feeling, conscious or not, that this era of monopoly may finally be over.

Of course, as a hacker I can't help thinking about how something broken could be fixed. Is there some way Microsoft could come back? In principle, yes. To see how, envision two things: (a) the amount of cash Microsoft now has on hand, and (b) Larry and Sergey making the rounds of all the search engines ten years ago trying to sell the idea for Google for a million dollars, and being turned down by everyone.

The surprising fact is, brilliant hackers—dangerously brilliant hackers—can be had very cheaply, by the standards of a company as rich as Microsoft. They can't hire smart people anymore, but they could buy as many as they wanted for only an order of magnitude more. So if they wanted to be a contender again, this is how they could do it:

1. Buy all the good "Web 2.0" startups. They could get substantially all of them for less than they'd have to pay for Facebook.

2. Put them all in a building in Silicon Valley, surrounded by lead shielding to protect them from any contact with Redmond.

I feel safe suggesting this, because they'd never do it. Microsoft's biggest weakness is that they still don't realize how much they suck. They still think they can write software in house. Maybe they can, by the standards of the desktop world. But that world ended a few years ago.

I already know what the reaction to this essay will be. Half the readers will say that Microsoft is still an enormously profitable company, and that I should be more careful about drawing conclusions based on what a few people think in our insular little "Web 2.0" bubble. The other half, the younger half, will complain that this is old news.

See also: Microsoft is Dead: the Cliffs Notes

Notes

[1] It doesn't take a conscious effort to make software incompatible. All you have to do is not work too hard at fixing bugs—which, if you're a big company, you produce in copious quantities. The situation is analogous to the writing of "literary theorists." Most don't try to be obscure; they just don't make an effort to be clear. It wouldn't pay.

[2] In part because Steve Jobs got pushed out by John Sculley in a way that's rare among technology companies. If Apple's board hadn't made that blunder, they wouldn't have had to bounce back.

Portuguese Translation

Simplified Chinese Translation

Korean Translation

Two Kinds of Judgement

April 2007

There are two different ways people judge you. Sometimes judging you correctly is the end goal. But there's a second much more common type of judgement where it isn't. We tend to regard all judgements of us as the first type. We'd probably be happier if we realized which are and which aren't.

The first type of judgement, the type where judging you is the end goal, include court cases, grades in classes, and most competitions. Such judgements can of course be mistaken, but because the goal is to judge you correctly, there's usually some kind of appeals process. If you feel you've been misjudged, you can protest that you've been treated unfairly.

Nearly all the judgements made on children are of this type, so we get into the habit early in life of thinking that all judgements are.

But in fact there is a second much larger class of judgements where judging you is only a means to something else. These include college admissions, hiring and investment decisions, and of course the judgements made in dating. This kind of judgement is not really about you.

Put yourself in the position of someone selecting players for a national team. Suppose for the sake of simplicity that this is a game with no positions, and that you have to select 20 players. There will be a few stars who clearly should make the team, and many players who clearly shouldn't. The only place your judgement makes a difference is in the borderline cases. Suppose you screw up and underestimate the 20th best player, causing him not to make the team, and his place to be taken by the 21st best. You've still picked a good team. If the players have the usual distribution of ability, the 21st best player will be only slightly worse than the 20th best. Probably the difference between them will be less than the measurement error.

The 20th best player may feel he has been misjudged. But your goal here wasn't to provide a service estimating people's ability. It was to pick a team, and if the difference between the 20th and 21st best players is less than the measurement error, you've still done that optimally.

It's a false analogy even to use the word unfair to describe this kind of

misjudgement. It's not aimed at producing a correct estimate of any given individual, but at selecting a reasonably optimal set.

One thing that leads us astray here is that the selector seems to be in a position of power. That makes him seem like a judge. If you regard someone judging you as a customer instead of a judge, the expectation of fairness goes away. The author of a good novel wouldn't complain that readers were *unfair* for preferring a potboiler with a racy cover. Stupid, perhaps, but not unfair.

Our early training and our self-centeredness combine to make us believe that every judgement of us is about us. In fact most aren't. This is a rare case where being less self-centered will make people more confident. Once you realize how little most people judging you care about judging you accurately—once you realize that because of the normal distribution of most applicant pools, it matters least to judge accurately in precisely the cases where judgement has the most effect—you won't take rejection so personally.

And curiously enough, taking rejection less personally may help you to get rejected less often. If you think someone judging you will work hard to judge you correctly, you can afford to be passive. But the more you realize that most judgements are greatly influenced by random, extraneous factors—that most people judging you are more like a fickle novel buyer than a wise and perceptive magistrate—the more you realize you can do things to influence the outcome.

One good place to apply this principle is in college applications. Most high school students applying to college do it with the usual child's mix of inferiority and self-centeredness: inferiority in that they assume that admissions committees must be all-seeing; self-centeredness in that they assume admissions committees care enough about them to dig down into their application and figure out whether they're good or not. These combine to make applicants passive in applying and hurt when they're rejected. If college applicants realized how quick and impersonal most selection processes are, they'd make more effort to sell themselves, and take the outcome less personally.

Spanish Translation

Russian Translation

Arabic Translation

The Hacker's Guide to Investors

April 2007

(This essay is derived from a keynote talk at the 2007 ASES Summit at Stanford.)

The world of investors is a foreign one to most hackers—partly because investors are so unlike hackers, and partly because they tend to operate in secret. I've been dealing with this world for many years, both as a founder and an investor, and I still don't fully understand it.

In this essay I'm going to list some of the more surprising things I've learned about investors. Some I only learned in the past year.

Teaching hackers how to deal with investors is probably the second most important thing we do at Y Combinator. The most important thing for a startup is to make something good. But everyone knows that's important. The dangerous thing about investors is that hackers don't know how little they know about this strange world.

1. The investors are what make a startup hub.

About a year ago I tried to figure out what you'd need to reproduce Silicon Valley. I decided the critical ingredients were rich people and nerds—investors and founders. People are all you need to make technology, and all the other people will move.

If I had to narrow that down, I'd say investors are the limiting factor. Not because they contribute more to the startup, but simply because they're least willing to move. They're rich. They're not going to move to Albuquerque just because there are some smart hackers there they could invest in. Whereas hackers will move to the Bay Area to find investors.

2. Angel investors are the most critical.

There are several types of investors. The two main categories are angels and VCs: VCs invest other people's money, and angels invest their own.

Though they're less well known, the angel investors are probably the more critical ingredient in creating a silicon valley. Most companies that VCs invest in would never have made it that far if angels hadn't invested first. VCs say between half

and three quarters of companies that raise series A rounds have taken some outside investment already. [1]

Angels are willing to fund riskier projects than VCs. They also give valuable advice, because (unlike VCs) many have been startup founders themselves.

Google's story shows the key role angels play. A lot of people know Google raised money from Kleiner and Sequoia. What most don't realize is how late. That VC round was a series B round; the premoney valuation was $75 million. Google was already a successful company at that point. Really, Google was funded with angel money.

It may seem odd that the canonical Silicon Valley startup was funded by angels, but this is not so surprising. Risk is always proportionate to reward. So the most successful startup of all is likely to have seemed an extremely risky bet at first, and that is exactly the kind VCs won't touch.

Where do angel investors come from? From other startups. So startup hubs like Silicon Valley benefit from something like the marketplace effect, but shifted in time: startups are there because startups were there.

3. Angels don't like publicity.

If angels are so important, why do we hear more about VCs? Because VCs like publicity. They need to market themselves to the investors who are their "customers"—the endowments and pension funds and rich families whose money they invest—and also to founders who might come to them for funding.

Angels don't need to market themselves to investors because they invest their own money. Nor do they want to market themselves to founders: they don't want random people pestering them with business plans. Actually, neither do VCs. Both angels and VCs get deals almost exclusively through personal introductions. [2]

The reason VCs want a strong brand is not to draw in more business plans over the transom, but so they win deals when competing against other VCs. Whereas angels are rarely in direct competition, because (a) they do fewer deals, (b) they're happy to split them, and (c) they invest at a point where the stream is broader.

4. Most investors, especially VCs, are not like founders.

Some angels are, or were, hackers. But most VCs are a different type of people: they're dealmakers.

If you're a hacker, here's a thought experiment you can run to understand why there are basically no hacker VCs: How would you like a job where you never got to make anything, but instead spent all your time listening to other people pitch (mostly terrible) projects, deciding whether to fund them, and sitting on their boards if you did? That would not be fun for most hackers. Hackers like to make

things. This would be like being an administrator.

Because most VCs are a different species of people from founders, it's hard to know what they're thinking. If you're a hacker, the last time you had to deal with these guys was in high school. Maybe in college you walked past their fraternity on your way to the lab. But don't underestimate them. They're as expert in their world as you are in yours. What they're good at is reading people, and making deals work to their advantage. Think twice before you try to beat them at that.

5. Most investors are momentum investors.

Because most investors are dealmakers rather than technology people, they generally don't understand what you're doing. I knew as a founder that most VCs didn't get technology. I also knew some made a lot of money. And yet it never occurred to me till recently to put those two ideas together and ask "How can VCs make money by investing in stuff they don't understand?"

The answer is that they're like momentum investors. You can (or could once) make a lot of money by noticing sudden changes in stock prices. When a stock jumps upward, you buy, and when it suddenly drops, you sell. In effect you're insider trading, without knowing what you know. You just know someone knows something, and that's making the stock move.

This is how most venture investors operate. They don't try to look at something and predict whether it will take off. They win by noticing that something *is* taking off a little sooner than everyone else. That generates almost as good returns as actually being able to pick winners. They may have to pay a little more than they would if they got in at the very beginning, but only a little.

Investors always say what they really care about is the team. Actually what they care most about is your traffic, then what other investors think, then the team. If you don't yet have any traffic, they fall back on number 2, what other investors think. And this, as you can imagine, produces wild oscillations in the "stock price" of a startup. One week everyone wants you, and they're begging not to be cut out of the deal. But all it takes is for one big investor to cool on you, and the next week no one will return your phone calls. We regularly have startups go from hot to cold or cold to hot in a matter of days, and literally nothing has changed.

There are two ways to deal with this phenomenon. If you're feeling really confident, you can try to ride it. You can start by asking a comparatively lowly VC for a small amount of money, and then after generating interest there, ask more prestigious VCs for larger amounts, stirring up a crescendo of buzz, and then "sell" at the top. This is extremely risky, and takes months even if you succeed. I wouldn't try it myself. My advice is to err on the side of safety: when someone offers you a decent deal, just take it and get on with building the company. Startups win or lose based on the quality of their product, not the quality of their funding deals.

6. Most investors are looking for big hits.

Venture investors like companies that could go public. That's where the big returns are. They know the odds of any individual startup going public are small, but they want to invest in those that at least have a *chance* of going public.

Currently the way VCs seem to operate is to invest in a bunch of companies, most of which fail, and one of which is Google. Those few big wins compensate for losses on their other investments. What this means is that most VCs will only invest in you if you're a potential Google. They don't care about companies that are a safe bet to be acquired for $20 million. There needs to be a chance, however small, of the company becoming really big.

Angels are different in this respect. They're happy to invest in a company where the most likely outcome is a $20 million acquisition if they can do it at a low enough valuation. But of course they like companies that could go public too. So having an ambitious long-term plan pleases everyone.

If you take VC money, you have to mean it, because the structure of VC deals prevents early acquisitions. If you take VC money, they won't let you sell early.

7. VCs want to invest large amounts.

The fact that they're running investment funds makes VCs want to invest large amounts. A typical VC fund is now hundreds of millions of dollars. If $400 million has to be invested by 10 partners, they have to invest $40 million each. VCs usually sit on the boards of companies they fund. If the average deal size was $1 million, each partner would have to sit on 40 boards, which would not be fun. So they prefer bigger deals, where they can put a lot of money to work at once.

VCs don't regard you as a bargain if you don't need a lot of money. That may even make you less attractive, because it means their investment creates less of a barrier to entry for competitors.

Angels are in a different position because they're investing their own money. They're happy to invest small amounts—sometimes as little as $20,000—as long as the potential returns look good enough. So if you're doing something inexpensive, go to angels.

8. Valuations are fiction.

VCs admit that valuations are an artifact. They decide how much money you need and how much of the company they want, and those two constraints yield a valuation.

Valuations increase as the size of the investment does. A company that an angel is willing to put $50,000 into at a valuation of a million can't take $6 million from VCs at that valuation. That would leave the founders less than a seventh of the

company between them (since the option pool would also come out of that seventh). Most VCs wouldn't want that, which is why you never hear of deals where a VC invests $6 million at a premoney valuation of $1 million.

If valuations change depending on the amount invested, that shows how far they are from reflecting any kind of value of the company.

Since valuations are made up, founders shouldn't care too much about them. That's not the part to focus on. In fact, a high valuation can be a bad thing. If you take funding at a premoney valuation of $10 million, you won't be selling the company for 20. You'll have to sell for over 50 for the VCs to get even a 5x return, which is low to them. More likely they'll want you to hold out for 100. But needing to get a high price decreases the chance of getting bought at all; many companies can buy you for $10 million, but only a handful for 100. And since a startup is like a pass/fail course for the founders, what you want to optimize is your chance of a good outcome, not the percentage of the company you keep.

So why do founders chase high valuations? They're tricked by misplaced ambition. They feel they've achieved more if they get a higher valuation. They usually know other founders, and if they get a higher valuation they can say "mine is bigger than yours." But funding is not the real test. The real test is the final outcome for the founder, and getting too high a valuation may just make a good outcome less likely.

The one advantage of a high valuation is that you get less dilution. But there is another less sexy way to achieve that: just take less money.

9. Investors look for founders like the current stars.

Ten years ago investors were looking for the next Bill Gates. This was a mistake, because Microsoft was a very anomalous startup. They started almost as a contract programming operation, and the reason they became huge was that IBM happened to drop the PC standard in their lap.

Now all the VCs are looking for the next Larry and Sergey. This is a good trend, because Larry and Sergey are closer to the ideal startup founders.

Historically investors thought it was important for a founder to be an expert in business. So they were willing to fund teams of MBAs who planned to use the money to pay programmers to build their product for them. This is like funding Steve Ballmer in the hope that the programmer he'll hire is Bill Gates—kind of backward, as the events of the Bubble showed. Now most VCs know they should be funding technical guys. This is more pronounced among the very top funds; the lamer ones still want to fund MBAs.

If you're a hacker, it's good news that investors are looking for Larry and Sergey. The bad news is, the only investors who can do it right are the ones who knew them when they were a couple of CS grad students, not the confident media stars

they are today. What investors still don't get is how clueless and tentative great founders can seem at the very beginning.

10. The contribution of investors tends to be underestimated.

Investors do more for startups than give them money. They're helpful in doing deals and arranging introductions, and some of the smarter ones, particularly angels, can give good advice about the product.

In fact, I'd say what separates the great investors from the mediocre ones is the quality of their advice. Most investors give advice, but the top ones give *good* advice.

Whatever help investors give a startup tends to be underestimated. It's to everyone's advantage to let the world think the founders thought of everything. The goal of the investors is for the company to become valuable, and the company seems more valuable if it seems like all the good ideas came from within.

This trend is compounded by the obsession that the press has with founders. In a company founded by two people, 10% of the ideas might come from the first guy they hire. Arguably they've done a bad job of hiring otherwise. And yet this guy will be almost entirely overlooked by the press.

I say this as a founder: the contribution of founders is always overestimated. The danger here is that new founders, looking at existing founders, will think that they're supermen that one couldn't possibly equal oneself. Actually they have a hundred different types of support people just offscreen making the whole show possible. [3]

11. VCs are afraid of looking bad.

I've been very surprised to discover how timid most VCs are. They seem to be afraid of looking bad to their partners, and perhaps also to the limited partners—the people whose money they invest.

You can measure this fear in how much less risk VCs are willing to take. You can tell they won't make investments for their fund that they might be willing to make themselves as angels. Though it's not quite accurate to say that VCs are less willing to take risks. They're less willing to do things that might look bad. That's not the same thing.

For example, most VCs would be very reluctant to invest in a startup founded by a pair of 18 year old hackers, no matter how brilliant, because if the startup failed their partners could turn on them and say "What, you invested $x million of our money in a pair of 18 year olds?" Whereas if a VC invested in a startup founded by three former banking executives in their 40s who planned to outsource their product development—which to my mind is actually a lot riskier than investing in a pair of really smart 18 year olds—he couldn't be faulted, if it failed, for making

such an apparently prudent investment.

As a friend of mine said, "Most VCs can't do anything that would sound bad to the kind of doofuses who run pension funds." Angels can take greater risks because they don't have to answer to anyone.

12. Being turned down by investors doesn't mean much.

Some founders are quite dejected when they get turned down by investors. They shouldn't take it so much to heart. To start with, investors are often wrong. It's hard to think of a successful startup that wasn't turned down by investors at some point. Lots of VCs rejected Google. So obviously the reaction of investors is not a very meaningful test.

Investors will often reject you for what seem to be superficial reasons. I read of one VC who turned down a startup simply because they'd given away so many little bits of stock that the deal required too many signatures to close. [4] The reason investors can get away with this is that they see so many deals. It doesn't matter if they underestimate you because of some surface imperfection, because the next best deal will be almost as good. Imagine picking out apples at a grocery store. You grab one with a little bruise. Maybe it's just a surface bruise, but why even bother checking when there are so many other unbruised apples to choose from?

Investors would be the first to admit they're often wrong. So when you get rejected by investors, don't think "we suck," but instead ask "do we suck?" Rejection is a question, not an answer.

13. Investors are emotional.

I've been surprised to discover how emotional investors can be. You'd expect them to be cold and calculating, or at least businesslike, but often they're not. I'm not sure if it's their position of power that makes them this way, or the large sums of money involved, but investment negotiations can easily turn personal. If you offend investors, they'll leave in a huff.

A while ago an eminent VC firm offered a series A round to a startup we'd seed funded. Then they heard a rival VC firm was also interested. They were so afraid that they'd be rejected in favor of this other firm that they gave the startup what's known as an "exploding termsheet." They had, I think, 24 hours to say yes or no, or the deal was off. Exploding termsheets are a somewhat dubious device, but not uncommon. What surprised me was their reaction when I called to talk about it. I asked if they'd still be interested in the startup if the rival VC didn't end up making an offer, and they said no. What rational basis could they have had for saying that? If they thought the startup was worth investing in, what difference should it make what some other VC thought? Surely it was their duty to their limited partners simply to invest in the best opportunities they found; they should be delighted if the other VC said no, because it would mean they'd overlooked a good opportunity.

But of course there was no rational basis for their decision. They just couldn't stand the idea of taking this rival firm's rejects.

In this case the exploding termsheet was not (or not only) a tactic to pressure the startup. It was more like the high school trick of breaking up with someone before they can break up with you. In an earlier essay I said that VCs were a lot like high school girls. A few VCs have joked about that characterization, but none have disputed it.

14. The negotiation never stops till the closing.

Most deals, for investment or acquisition, happen in two phases. There's an initial phase of negotiation about the big questions. If this succeeds you get a termsheet, so called because it outlines the key terms of a deal. A termsheet is not legally binding, but it is a definite step. It's supposed to mean that a deal is going to happen, once the lawyers work out all the details. In theory these details are minor ones; by definition all the important points are supposed to be covered in the termsheet.

Inexperience and wishful thinking combine to make founders feel that when they have a termsheet, they have a deal. They want there to be a deal; everyone acts like they have a deal; so there must be a deal. But there isn't and may not be for several months. A lot can change for a startup in several months. It's not uncommon for investors and acquirers to get buyer's remorse. So you have to keep pushing, keep selling, all the way to the close. Otherwise all the "minor" details left unspecified in the termsheet will be interpreted to your disadvantage. The other side may even break the deal; if they do that, they'll usually seize on some technicality or claim you misled them, rather than admitting they changed their minds.

It can be hard to keep the pressure on an investor or acquirer all the way to the closing, because the most effective pressure is competition from other investors or acquirers, and these tend to drop away when you get a termsheet. You should try to stay as close friends as you can with these rivals, but the most important thing is just to keep up the momentum in your startup. The investors or acquirers chose you because you seemed hot. Keep doing whatever made you seem hot. Keep releasing new features; keep getting new users; keep getting mentioned in the press and in blogs.

15. Investors like to co-invest.

I've been surprised how willing investors are to split deals. You might think that if they found a good deal they'd want it all to themselves, but they seem positively eager to syndicate. This is understandable with angels; they invest on a smaller scale and don't like to have too much money tied up in any one deal. But VCs also share deals a lot. Why?

Partly I think this is an artifact of the rule I quoted earlier: after traffic, VCs care

most what other VCs think. A deal that has multiple VCs interested in it is more likely to close, so of deals that close, more will have multiple investors.

There is one rational reason to want multiple VCs in a deal: Any investor who co-invests with you is one less investor who could fund a competitor. Apparently Kleiner and Sequoia didn't like splitting the Google deal, but it did at least have the advantage, from each one's point of view, that there probably wouldn't be a competitor funded by the other. Splitting deals thus has similar advantages to confusing paternity.

But I think the main reason VCs like splitting deals is the fear of looking bad. If another firm shares the deal, then in the event of failure it will seem to have been a prudent choice—a consensus decision, rather than just the whim of an individual partner.

16. Investors collude.

Investing is not covered by antitrust law. At least, it better not be, because investors regularly do things that would be illegal otherwise. I know personally of cases where one investor has talked another out of making a competitive offer, using the promise of sharing future deals.

In principle investors are all competing for the same deals, but the spirit of cooperation is stronger than the spirit of competition. The reason, again, is that there are so many deals. Though a professional investor may have a closer relationship with a founder he invests in than with other investors, his relationship with the founder is only going to last a couple years, whereas his relationship with other firms will last his whole career. There isn't so much at stake in his interactions with other investors, but there will be a lot of them. Professional investors are constantly trading little favors.

Another reason investors stick together is to preserve the power of investors as a whole. So you will not, as of this writing, be able to get investors into an auction for your series A round. They'd rather lose the deal than establish a precedent of VCs competitively bidding against one another. An efficient startup funding market may be coming in the distant future; things tend to move in that direction; but it's certainly not here now.

17. Large-scale investors care about their portfolio, not any individual company.

The reason startups work so well is that everyone with power also has equity. The only way any of them can succeed is if they all do. This makes everyone naturally pull in the same direction, subject to differences of opinion about tactics.

The problem is, larger scale investors don't have exactly the same motivation. Close, but not identical. They don't need any given startup to succeed, like founders do, just their portfolio as a whole to. So in borderline cases the rational

thing for them to do is to sacrifice unpromising startups.

Large-scale investors tend to put startups in three categories: successes, failures, and the "living dead"—companies that are plugging along but don't seem likely in the immediate future to get bought or go public. To the founders, "living dead" sounds harsh. These companies may be far from failures by ordinary standards. But they might as well be from a venture investor's point of view, and they suck up just as much time and attention as the successes. So if such a company has two possible strategies, a conservative one that's slightly more likely to work in the end, or a risky one that within a short time will either yield a giant success or kill the company, VCs will push for the kill-or-cure option. To them the company is already a write-off. Better to have resolution, one way or the other, as soon as possible.

If a startup gets into real trouble, instead of trying to save it VCs may just sell it at a low price to another of their portfolio companies. Philip Greenspun said in *Founders at Work* that Ars Digita's VCs did this to them.

18. Investors have different risk profiles from founders.

Most people would rather a 100% chance of $1 million than a 20% chance of $10 million. Investors are rich enough to be rational and prefer the latter. So they'll always tend to encourage founders to keep rolling the dice. If a company is doing well, investors will want founders to turn down most acquisition offers. And indeed, most startups that turn down acquisition offers ultimately do better. But it's still hair-raising for the founders, because they might end up with nothing. When someone's offering to buy you for a price at which your stock is worth $5 million, saying no is equivalent to having $5 million and betting it all on one spin of the roulette wheel.

Investors will tell you the company is worth more. And they may be right. But that doesn't mean it's wrong to sell. Any financial advisor who put all his client's assets in the stock of a single, private company would probably lose his license for it.

More and more, investors are letting founders cash out partially. That should correct the problem. Most founders have such low standards that they'll feel rich with a sum that doesn't seem huge to investors. But this custom is spreading too slowly, because VCs are afraid of seeming irresponsible. No one wants to be the first VC to give someone fuck-you money and then actually get told "fuck you." But until this does start to happen, we know VCs are being too conservative.

19. Investors vary greatly.

Back when I was a founder I used to think all VCs were the same. And in fact they do all look the same. They're all what hackers call "suits." But since I've been dealing with VCs more I've learned that some suits are smarter than others.

They're also in a business where winners tend to keep winning and losers to keep

losing. When a VC firm has been successful in the past, everyone wants funding from them, so they get the pick of all the new deals. The self-reinforcing nature of the venture funding market means that the top ten firms live in a completely different world from, say, the hundredth. As well as being smarter, they tend to be calmer and more upstanding; they don't need to do iffy things to get an edge, and don't want to because they have more brand to protect.

There are only two kinds of VCs you want to take money from, if you have the luxury of choosing: the "top tier" VCs, meaning about the top 20 or so firms, plus a few new ones that are not among the top 20 only because they haven't been around long enough.

It's particularly important to raise money from a top firm if you're a hacker, because they're more confident. That means they're less likely to stick you with a business guy as CEO, like VCs used to do in the 90s. If you seem smart and want to do it, they'll let you run the company.

20. Investors don't realize how much it costs to raise money from them.

Raising money is a huge time suck at just the point where startups can least afford it. It's not unusual for it to take five or six months to close a funding round. Six weeks is fast. And raising money is not just something you can leave running as a background process. When you're raising money, it's inevitably the main focus of the company. Which means building the product isn't.

Suppose a Y Combinator company starts talking to VCs after demo day, and is successful in raising money from them, closing the deal after a comparatively short 8 weeks. Since demo day occurs after 10 weeks, the company is now 18 weeks old. Raising money, rather than working on the product, has been the company's main focus for 44% of its existence. And mind you, this an example where things turned out *well*.

When a startup does return to working on the product after a funding round finally closes, it's as if they were returning to work after a months-long illness. They've lost most of their momentum.

Investors have no idea how much they damage the companies they invest in by taking so long to do it. But companies do. So there is a big opportunity here for a new kind of venture fund that invests smaller amounts at lower valuations, but promises to either close or say no very quickly. If there were such a firm, I'd recommend it to startups in preference to any other, no matter how prestigious. Startups live on speed and momentum.

21. Investors don't like to say no.

The reason funding deals take so long to close is mainly that investors can't make up their minds. VCs are not big companies; they can do a deal in 24 hours if they need to. But they usually let the initial meetings stretch out over a couple weeks.

The reason is the selection algorithm I mentioned earlier. Most don't try to predict whether a startup will win, but to notice quickly that it already is winning. They care what the market thinks of you and what other VCs think of you, and they can't judge those just from meeting you.

Because they're investing in things that (a) change fast and (b) they don't understand, a lot of investors will reject you in a way that can later be claimed not to have been a rejection. Unless you know this world, you may not even realize you've been rejected. Here's a VC saying no:

> We're really excited about your project, and we want to keep in close touch as you develop it further.

Translated into more straightforward language, this means: We're not investing in you, but we may change our minds if it looks like you're taking off. Sometimes they're more candid and say explicitly that they need to "see some traction." They'll invest in you if you start to get lots of users. But so would any VC. So all they're saying is that you're still at square 1.

Here's a test for deciding whether a VC's response was yes or no. Look down at your hands. Are you holding a termsheet?

22. You need investors.

Some founders say "Who needs investors?" Empirically the answer seems to be: everyone who wants to succeed. Practically every successful startup takes outside investment at some point.

Why? What the people who think they don't need investors forget is that they will have competitors. The question is not whether you *need* outside investment, but whether it could help you at all. If the answer is yes, and you don't take investment, then competitors who do will have an advantage over you. And in the startup world a little advantage can expand into a lot.

Mike Moritz famously said that he invested in Yahoo because he thought they had a few weeks' lead over their competitors. That may not have mattered quite so much as he thought, because Google came along three years later and kicked Yahoo's ass. But there is something in what he said. Sometimes a small lead can grow into the yes half of a binary choice.

Maybe as it gets cheaper to start a startup, it will start to be possible to succeed in a competitive market without outside funding. There are certainly costs to raising money. But as of this writing the empirical evidence says it's a net win.

23. Investors like it when you don't need them.

A lot of founders approach investors as if they needed their permission to start a company—as if it were like getting into college. But you don't need investors to

start most companies; they just make it easier.

And in fact, investors greatly prefer it if you don't need them. What excites them, both consciously and unconsciously, is the sort of startup that approaches them saying "the train's leaving the station; are you in or out?" not the one saying "please can we have some money to start a company?"

Most investors are "bottoms" in the sense that the startups they like most are those that are rough with them. When Google stuck Kleiner and Sequoia with a $75 million premoney valuation, their reaction was probably "Ouch! That feels so good." And they were right, weren't they? That deal probably made them more than any other they've done.

The thing is, VCs are pretty good at reading people. So don't try to act tough with them unless you really are the next Google, or they'll see through you in a second. Instead of acting tough, what most startups should do is simply always have a backup plan. Always have some alternative plan for getting started if any given investor says no. Having one is the best insurance against needing one.

So you shouldn't start a startup that's expensive to start, because then you'll be at the mercy of investors. If you ultimately want to do something that will cost a lot, start by doing a cheaper subset of it, and expand your ambitions when and if you raise more money.

Apparently the most likely animals to be left alive after a nuclear war are cockroaches, because they're so hard to kill. That's what you want to be as a startup, initially. Instead of a beautiful but fragile flower that needs to have its stem in a plastic tube to support itself, better to be small, ugly, and indestructible.

Notes

[1] I may be underestimating VCs. They may play some behind the scenes role in IPOs, which you ultimately need if you want to create a silicon valley.

[2] A few VCs have an email address you can send your business plan to, but the number of startups that get funded this way is basically zero. You should always get a personal introduction—and to a partner, not an associate.

[3] Several people have told us that the most valuable thing about startup school was that they got to see famous startup founders and realized they were just ordinary guys. Though we're happy to provide this service, this is not generally the way we pitch startup school to potential speakers.

[4] Actually this sounds to me like a VC who got buyer's remorse, then used a

technicality to get out of the deal. But it's telling that it even seemed a plausible excuse.

Thanks to Sam Altman, Paul Buchheit, Hutch Fishman, and Robert Morris for reading drafts of this, and to Kenneth King of ASES for inviting me to speak.

 Comment on this essay.

An Alternative Theory of Unions

May 2007

People who worry about the increasing gap between rich and poor generally look back on the mid twentieth century as a golden age. In those days we had a large number of high-paying union manufacturing jobs that boosted the median income. I wouldn't quite call the high-paying union job a myth, but I think people who dwell on it are reading too much into it.

Oddly enough, it was working with startups that made me realize where the high-paying union job came from. In a rapidly growing market, you don't worry too much about efficiency. It's more important to grow fast. If there's some mundane problem getting in your way, and there's a simple solution that's somewhat expensive, just take it and get on with more important things. EBay didn't win by paying less for servers than their competitors.

Difficult though it may be to imagine now, manufacturing was a growth industry in the mid twentieth century. This was an era when small firms making everything from cars to candy were getting consolidated into a new kind of corporation with national reach and huge economies of scale. You had to grow fast or die. Workers were for these companies what servers are for an Internet startup. A reliable supply was more important than low cost.

If you looked in the head of a 1950s auto executive, the attitude must have been: sure, give 'em whatever they ask for, so long as the new model isn't delayed.

In other words, those workers were not paid what their work was worth. Circumstances being what they were, companies would have been stupid to insist on paying them so little.

If you want a less controversial example of this phenomenon, ask anyone who worked as a consultant building web sites during the Internet Bubble. In the late nineties you could get paid huge sums of money for building the most trivial

things. And yet does anyone who was there have any expectation those days will ever return? I doubt it. Surely everyone realizes that was just a temporary aberration.

The era of labor unions seems to have been the same kind of aberration, just spread over a longer period, and mixed together with a lot of ideology that prevents people from viewing it with as cold an eye as they would something like consulting during the Bubble.

Basically, unions were just Razorfish.

People who think the labor movement was the creation of heroic union organizers have a problem to explain: why are unions shrinking now? The best they can do is fall back on the default explanation of people living in fallen civilizations. Our ancestors were giants. The workers of the early twentieth century must have had a moral courage that's lacking today.

In fact there's a simpler explanation. The early twentieth century was just a fast-growing startup overpaying for infrastructure. And we in the present are not a fallen people, who have abandoned whatever mysterious high-minded principles produced the high-paying union job. We simply live in a time when the fast-growing companies overspend on different things.

The Equity Equation

July 2007

An investor wants to give you money for a certain percentage of your startup. Should you take it? You're about to hire your first employee. How much stock should you give him?

These are some of the hardest questions founders face. And yet both have the same answer:

1/(1 - n)

Whenever you're trading stock in your company for anything, whether it's money or an employee or a deal with another company, the test for whether to do it is the same. You should give up n% of your company if what you trade it for improves your average outcome enough that the (100 - n)% you have left is worth more than the whole company was before.

For example, if an investor wants to buy half your company, how much does that investment have to improve your average outcome for you to break even? Obviously it has to double: if you trade half your company for something that more than doubles the company's average outcome, you're net ahead. You have half as big a share of something worth more than twice as much.

In the general case, if n is the fraction of the company you're giving up, the deal is a good one if it makes the company worth more than 1/(1 - n).

For example, suppose Y Combinator offers to fund you in return for 7% of your company. In this case, n is .07 and 1/(1 - n) is 1.075. So you should take the deal if you believe we can improve your average outcome by more than 7.5%. If we improve your outcome by 10%, you're net ahead, because the remaining .93 you hold is worth .93 x 1.1 = 1.023. [1]

One of the things the equity equation shows us is that, financially at least, taking money from a top VC firm can be a really good deal. Greg Mcadoo from Sequoia recently said at a YC dinner that when Sequoia invests alone they like to take about 30% of a company. 1/.7 = 1.43, meaning that deal is worth taking if they can improve your outcome by more than 43%. For the average startup, that would be an extraordinary bargain. It would improve the average startup's prospects by

more than 43% just to be able to *say* they were funded by Sequoia, even if they never actually got the money.

The reason Sequoia is such a good deal is that the percentage of the company they take is artificially low. They don't even try to get market price for their investment; they limit their holdings to leave the founders enough stock to feel the company is still theirs.

The catch is that Sequoia gets about 6000 business plans a year and funds about 20 of them, so the odds of getting this great deal are 1 in 300. The companies that make it through are not average startups.

Of course, there are other factors to consider in a VC deal. It's never just a straight trade of money for stock. But if it were, taking money from a top firm would generally be a bargain.

You can use the same formula when giving stock to employees, but it works in the other direction. If i is the average outcome for the company with the addition of some new person, then they're worth n such that $i = 1/(1 - n)$. Which means $n = (i - 1)/i$.

For example, suppose you're just two founders and you want to hire an additional hacker who's so good you feel he'll increase the average outcome of the whole company by 20%. $n = (1.2 - 1)/1.2 = .167$. So you'll break even if you trade 16.7% of the company for him.

That doesn't mean 16.7% is the right amount of stock to give him. Stock is not the only cost of hiring someone: there's usually salary and overhead as well. And if the company merely breaks even on the deal, there's no reason to do it.

I think to translate salary and overhead into stock you should multiply the annual rate by about 1.5. Most startups grow fast or die; if you die you don't have to pay the guy, and if you grow fast you'll be paying next year's salary out of next year's valuation, which should be 3x this year's. If your valuation grows 3x a year, the total cost in stock of a new hire's salary and overhead is 1.5 years' cost at the present valuation. [2]

How much of an additional margin should the company need as the "activation energy" for the deal? Since this is in effect the company's profit on a hire, the market will determine that: if you're a hot opportunity, you can charge more.

Let's run through an example. Suppose the company wants to make a "profit" of 50% on the new hire mentioned above. So subtract a third from 16.7% and we have 11.1% as his "retail" price. Suppose further that he's going to cost $60k a year in salary and overhead, x 1.5 = $90k total. If the company's valuation is $2 million, $90k is 4.5%. 11.1% - 4.5% = an offer of 6.6%.

Incidentally, notice how important it is for early employees to take little salary. It

comes right out of stock that could otherwise be given to them.

Obviously there is a great deal of play in these numbers. I'm not claiming that stock grants can now be reduced to a formula. Ultimately you always have to guess. But at least know what you're guessing. If you choose a number based on your gut feel, or a table of typical grant sizes supplied by a VC firm, understand what those are estimates of.

And more generally, when you make any decision involving equity, run it through 1/(1 - n) to see if it makes sense. You should always feel richer after trading equity. If the trade didn't increase the value of your remaining shares enough to put you net ahead, you wouldn't have (or shouldn't have) done it.

Notes

[1] This is why we can't believe anyone would think Y Combinator was a bad deal. Does anyone really think we're so useless that in three months we can't improve a startup's prospects by 7.5%?

[2] The obvious choice for your present valuation is the post-money valuation of your last funding round. This probably undervalues the company, though, because (a) unless your last round just happened, the company is presumably worth more, and (b) the valuation of an early funding round usually reflects some other contribution by the investors.

Thanks to Sam Altman, Trevor Blackwell, Paul Buchheit, Hutch Fishman, David Hornik, Paul Kedrosky, Jessica Livingston, Gary Sabot, and Joshua Schachter for reading drafts of this.

Stuff

July 2007

I have too much stuff. Most people in America do. In fact, the poorer people are, the more stuff they seem to have. Hardly anyone is so poor that they can't afford a front yard full of old cars.

It wasn't always this way. Stuff used to be rare and valuable. You can still see evidence of that if you look for it. For example, in my house in Cambridge, which was built in 1876, the bedrooms don't have closets. In those days people's stuff fit in a chest of drawers. Even as recently as a few decades ago there was a lot less stuff. When I look back at photos from the 1970s, I'm surprised how empty houses look. As a kid I had what I thought was a huge fleet of toy cars, but they'd be dwarfed by the number of toys my nephews have. All together my Matchboxes and Corgis took up about a third of the surface of my bed. In my nephews' rooms the bed is the only clear space.

Stuff has gotten a lot cheaper, but our attitudes toward it haven't changed correspondingly. We overvalue stuff.

That was a big problem for me when I had no money. I felt poor, and stuff seemed valuable, so almost instinctively I accumulated it. Friends would leave something behind when they moved, or I'd see something as I was walking down the street on trash night (beware of anything you find yourself describing as "perfectly good"), or I'd find something in almost new condition for a tenth its retail price at a garage sale. And pow, more stuff.

In fact these free or nearly free things weren't bargains, because they were worth even less than they cost. Most of the stuff I accumulated was worthless, because I didn't need it.

What I didn't understand was that the value of some new acquisition wasn't the difference between its retail price and what I paid for it. It was the value I derived from it. Stuff is an extremely illiquid asset. Unless you have some plan for selling that valuable thing you got so cheaply, what difference does it make what it's "worth?" The only way you're ever going to extract any value from it is to use it. And if you don't have any immediate use for it, you probably never will.

Companies that sell stuff have spent huge sums training us to think stuff is still

valuable. But it would be closer to the truth to treat stuff as worthless.

In fact, worse than worthless, because once you've accumulated a certain amount of stuff, it starts to own you rather than the other way around. I know of one couple who couldn't retire to the town they preferred because they couldn't afford a place there big enough for all their stuff. Their house isn't theirs; it's their stuff's.

And unless you're extremely organized, a house full of stuff can be very depressing. A cluttered room saps one's spirits. One reason, obviously, is that there's less room for people in a room full of stuff. But there's more going on than that. I think humans constantly scan their environment to build a mental model of what's around them. And the harder a scene is to parse, the less energy you have left for conscious thoughts. A cluttered room is literally exhausting.

(This could explain why clutter doesn't seem to bother kids as much as adults. Kids are less perceptive. They build a coarser model of their surroundings, and this consumes less energy.)

I first realized the worthlessness of stuff when I lived in Italy for a year. All I took with me was one large backpack of stuff. The rest of my stuff I left in my landlady's attic back in the US. And you know what? All I missed were some of the books. By the end of the year I couldn't even remember what else I had stored in that attic.

And yet when I got back I didn't discard so much as a box of it. Throw away a perfectly good rotary telephone? I might need that one day.

The really painful thing to recall is not just that I accumulated all this useless stuff, but that I often spent money I desperately needed on stuff that I didn't.

Why would I do that? Because the people whose job is to sell you stuff are really, really good at it. The average 25 year old is no match for companies that have spent years figuring out how to get you to spend money on stuff. They make the experience of buying stuff so pleasant that "shopping" becomes a leisure activity.

How do you protect yourself from these people? It can't be easy. I'm a fairly skeptical person, and their tricks worked on me well into my thirties. But one thing that might work is to ask yourself, before buying something, "is this going to make my life noticeably better?"

A friend of mine cured herself of a clothes buying habit by asking herself before she bought anything "Am I going to wear this all the time?" If she couldn't convince herself that something she was thinking of buying would become one of those few things she wore all the time, she wouldn't buy it. I think that would work for any kind of purchase. Before you buy anything, ask yourself: will this be something I use constantly? Or is it just something nice? Or worse still, a mere bargain?

The worst stuff in this respect may be stuff you don't use much because it's too good. Nothing owns you like fragile stuff. For example, the "good china" so many households have, and whose defining quality is not so much that it's fun to use, but that one must be especially careful not to break it.

Another way to resist acquiring stuff is to think of the overall cost of owning it. The purchase price is just the beginning. You're going to have to *think* about that thing for years—perhaps for the rest of your life. Every thing you own takes energy away from you. Some give more than they take. Those are the only things worth having.

I've now stopped accumulating stuff. Except books—but books are different. Books are more like a fluid than individual objects. It's not especially inconvenient to own several thousand books, whereas if you owned several thousand random possessions you'd be a local celebrity. But except for books, I now actively avoid stuff. If I want to spend money on some kind of treat, I'll take services over goods any day.

I'm not claiming this is because I've achieved some kind of zenlike detachment from material things. I'm talking about something more mundane. A historical change has taken place, and I've now realized it. Stuff used to be valuable, and now it's not.

In industrialized countries the same thing happened with food in the middle of the twentieth century. As food got cheaper (or we got richer; they're indistinguishable), eating too much started to be a bigger danger than eating too little. We've now reached that point with stuff. For most people, rich or poor, stuff has become a burden.

The good news is, if you're carrying a burden without knowing it, your life could be better than you realize. Imagine walking around for years with five pound ankle weights, then suddenly having them removed.

Spanish Translation

Russian Translation

Italian Translation

Polish Translation

Turkish Translation

French Translation

Slovak Translation

Romanian Translation

German Translation

Holding a Program in One's Head

August 2007

A good programmer working intensively on his own code can hold it in his mind the way a mathematician holds a problem he's working on. Mathematicians don't answer questions by working them out on paper the way schoolchildren are taught to. They do more in their heads: they try to understand a problem space well enough that they can walk around it the way you can walk around the memory of the house you grew up in. At its best programming is the same. You hold the whole program in your head, and you can manipulate it at will.

That's particularly valuable at the start of a project, because initially the most important thing is to be able to change what you're doing. Not just to solve the problem in a different way, but to change the problem you're solving.

Your code is your understanding of the problem you're exploring. So it's only when you have your code in your head that you really understand the problem.

It's not easy to get a program into your head. If you leave a project for a few months, it can take days to really understand it again when you return to it. Even when you're actively working on a program it can take half an hour to load into your head when you start work each day. And that's in the best case. Ordinary programmers working in typical office conditions never enter this mode. Or to put it more dramatically, ordinary programmers working in typical office conditions never really understand the problems they're solving.

Even the best programmers don't always have the whole program they're working on loaded into their heads. But there are things you can do to help:

1. **Avoid distractions.** Distractions are bad for many types of work, but especially bad for programming, because programmers tend to operate at the limit of the detail they can handle.

   The danger of a distraction depends not on how long it is, but on how much it scrambles your brain. A programmer can leave the office and go and get a sandwich without losing the code in his head. But the wrong kind of interruption can wipe your brain in 30 seconds.

   Oddly enough, scheduled distractions may be worse than unscheduled ones. If you know you have a meeting in an hour, you don't even start working on

something hard.

2. **Work in long stretches.** Since there's a fixed cost each time you start working on a program, it's more efficient to work in a few long sessions than many short ones. There will of course come a point where you get stupid because you're tired. This varies from person to person. I've heard of people hacking for 36 hours straight, but the most I've ever been able to manage is about 18, and I work best in chunks of no more than 12.

   The optimum is not the limit you can physically endure. There's an advantage as well as a cost of breaking up a project. Sometimes when you return to a problem after a rest, you find your unconscious mind has left an answer waiting for you.

3. **Use succinct languages.** More powerful programming languages make programs shorter. And programmers seem to think of programs at least partially in the language they're using to write them. The more succinct the language, the shorter the program, and the easier it is to load and keep in your head.

   You can magnify the effect of a powerful language by using a style called bottom-up programming, where you write programs in multiple layers, the lower ones acting as programming languages for those above. If you do this right, you only have to keep the topmost layer in your head.

4. **Keep rewriting your program.** Rewriting a program often yields a cleaner design. But it would have advantages even if it didn't: you have to understand a program completely to rewrite it, so there is no better way to get one loaded into your head.

5. **Write rereadable code.** All programmers know it's good to write readable code. But you yourself are the most important reader. Especially in the beginning; a prototype is a conversation with yourself. And when writing for yourself you have different priorities. If you're writing for other people, you may not want to make code too dense. Some parts of a program may be easiest to read if you spread things out, like an introductory textbook. Whereas if you're writing code to make it easy to reload into your head, it may be best to go for brevity.

6. **Work in small groups.** When you manipulate a program in your head, your vision tends to stop at the edge of the code you own. Other parts you don't understand as well, and more importantly, can't take liberties with. So the smaller the number of programmers, the more completely a project can mutate. If there's just one programmer, as there often is at first, you can do all-encompassing redesigns.

7. **Don't have multiple people editing the same piece of code.** You never understand other people's code as well as your own. No matter how thoroughly you've read it, you've only read it, not written it. So if a piece of code is written by multiple authors, none of them understand it as well as a single author would.

   And of course you can't safely redesign something other people are working on. It's not just that you'd have to ask permission. You don't even let

yourself think of such things. Redesigning code with several authors is like changing laws; redesigning code you alone control is like seeing the other interpretation of an ambiguous image.

If you want to put several people to work on a project, divide it into components and give each to one person.

8. **Start small.** A program gets easier to hold in your head as you become familiar with it. You can start to treat parts as black boxes once you feel confident you've fully explored them. But when you first start working on a project, you're forced to see everything. If you start with too big a problem, you may never quite be able to encompass it. So if you need to write a big, complex program, the best way to begin may not be to write a spec for it, but to write a prototype that solves a subset of the problem. Whatever the advantages of planning, they're often outweighed by the advantages of being able to keep a program in your head.

It's striking how often programmers manage to hit all eight points by accident. Someone has an idea for a new project, but because it's not officially sanctioned, he has to do it in off hours—which turn out to be more productive because there are no distractions. Driven by his enthusiasm for the new project he works on it for many hours at a stretch. Because it's initially just an experiment, instead of a "production" language he uses a mere "scripting" language—which is in fact far more powerful. He completely rewrites the program several times; that wouldn't be justifiable for an official project, but this is a labor of love and he wants it to be perfect. And since no one is going to see it except him, he omits any comments except the note-to-self variety. He works in a small group perforce, because he either hasn't told anyone else about the idea yet, or it seems so unpromising that no one else is allowed to work on it. Even if there is a group, they couldn't have multiple people editing the same code, because it changes too fast for that to be possible. And the project starts small because the idea *is* small at first; he just has some cool hack he wants to try out.

Even more striking are the number of officially sanctioned projects that manage to do *all eight things wrong*. In fact, if you look at the way software gets written in most organizations, it's almost as if they were deliberately trying to do things wrong. In a sense, they are. One of the defining qualities of organizations since there have been such a thing is to treat individuals as interchangeable parts. This works well for more parallelizable tasks, like fighting wars. For most of history a well-drilled army of professional soldiers could be counted on to beat an army of individual warriors, no matter how valorous. But having ideas is not very parallelizable. And that's what programs are: ideas.

It's not merely true that organizations dislike the idea of depending on individual genius, it's a tautology. It's part of the definition of an organization not to. Of our current concept of an organization, at least.

Maybe we could define a new kind of organization that combined the efforts of individuals without requiring them to be interchangeable. Arguably a market is such a form of organization, though it may be more accurate to describe a market

as a degenerate case—as what you get by default when organization isn't possible.

Probably the best we'll do is some kind of hack, like making the programming parts of an organization work differently from the rest. Perhaps the optimal solution is for big companies not even to try to develop ideas in house, but simply to buy them. But regardless of what the solution turns out to be, the first step is to realize there's a problem. There is a contradiction in the very phrase "software company." The two words are pulling in opposite directions. Any good programmer in a large organization is going to be at odds with it, because organizations are designed to prevent what programmers strive for.

Good programmers manage to get a lot done anyway. But often it requires practically an act of rebellion against the organizations that employ them. Perhaps it will help if more people understand that the way programmers behave is driven by the demands of the work they do. It's not because they're irresponsible that they work in long binges during which they blow off all other obligations, plunge straight into programming instead of writing specs first, and rewrite code that already works. It's not because they're unfriendly that they prefer to work alone, or growl at people who pop their head in the door to say hello. This apparently random collection of annoying habits has a single explanation: the power of holding a program in one's head.

Whether or not understanding this can help large organizations, it can certainly help their competitors. The weakest point in big companies is that they don't let individual programmers do great work. So if you're a little startup, this is the place to attack them. Take on the kind of problems that have to be solved in one big brain.

Thanks to Sam Altman, David Greenspan, Aaron Iba, Jessica Livingston, Robert Morris, Peter Norvig, Lisa Randall, Emmett Shear, Sergei Tsarev, and Stephen Wolfram for reading drafts of this.

Japanese Translation

Simplified Chinese Translation

Portuguese Translation

Bulgarian Translation

Russian Translation

How Not to Die

August 2007

(This is a talk I gave at the last Y Combinator dinner of the summer. Usually we don't have a speaker at the last dinner; it's more of a party. But it seemed worth spoiling the atmosphere if I could save some of the startups from preventable deaths. So at the last minute I cooked up this rather grim talk. I didn't mean this as an essay; I wrote it down because I only had two hours before dinner and think fastest while writing.)

A couple days ago I told a reporter that we expected about a third of the companies we funded to succeed. Actually I was being conservative. I'm hoping it might be as much as a half. Wouldn't it be amazing if we could achieve a 50% success rate?

Another way of saying that is that half of you are going to die. Phrased that way, it doesn't sound good at all. In fact, it's kind of weird when you think about it, because our definition of success is that the founders get rich. If half the startups we fund succeed, then half of you are going to get rich and the other half are going to get nothing.

If you can just avoid dying, you get rich. That sounds like a joke, but it's actually a pretty good description of what happens in a typical startup. It certainly describes what happened in Viaweb. We avoided dying till we got rich.

It was really close, too. When we were visiting Yahoo to talk about being acquired, we had to interrupt everything and borrow one of their conference rooms to talk down an investor who was about to back out of a new funding round we needed to stay alive. So even in the middle of getting rich we were fighting off the grim reaper.

You may have heard that quote about luck consisting of opportunity meeting preparation. You've now done the preparation. The work you've done so far has, in effect, put you in a position to get lucky: you can now get rich by not letting your company die. That's more than most people have. So let's talk about how not to die.

We've done this five times now, and we've seen a bunch of startups die. About 10

of them so far. We don't know exactly what happens when they die, because they generally don't die loudly and heroically. Mostly they crawl off somewhere and die.

For us the main indication of impending doom is when we don't hear from you. When we haven't heard from, or about, a startup for a couple months, that's a bad sign. If we send them an email asking what's up, and they don't reply, that's a really bad sign. So far that is a 100% accurate predictor of death.

Whereas if a startup regularly does new deals and releases and either sends us mail or shows up at YC events, they're probably going to live.

I realize this will sound naive, but maybe the linkage works in both directions. Maybe if you can arrange that we keep hearing from you, you won't die.

That may not be so naive as it sounds. You've probably noticed that having dinners every Tuesday with us and the other founders causes you to get more done than you would otherwise, because every dinner is a mini Demo Day. Every dinner is a kind of a deadline. So the mere constraint of staying in regular contact with us will push you to make things happen, because otherwise you'll be embarrassed to tell us that you haven't done anything new since the last time we talked.

If this works, it would be an amazing hack. It would be pretty cool if merely by staying in regular contact with us you could get rich. It sounds crazy, but there's a good chance that would work.

A variant is to stay in touch with other YC-funded startups. There is now a whole neighborhood of them in San Francisco. If you move there, the peer pressure that made you work harder all summer will continue to operate.

When startups die, the official cause of death is always either running out of money or a critical founder bailing. Often the two occur simultaneously. But I think the underlying cause is usually that they've become demoralized. You rarely hear of a startup that's working around the clock doing deals and pumping out new features, and dies because they can't pay their bills and their ISP unplugs their server.

Startups rarely die in mid keystroke. So keep typing!

If so many startups get demoralized and fail when merely by hanging on they could get rich, you have to assume that running a startup can be demoralizing. That is certainly true. I've been there, and that's why I've never done another startup. The low points in a startup are just unbelievably low. I bet even Google had moments where things seemed hopeless.

Knowing that should help. If you know it's going to feel terrible sometimes, then when it feels terrible you won't think "ouch, this feels terrible, I give up." It feels that way for everyone. And if you just hang on, things will probably get better. The metaphor people use to describe the way a startup feels is at least a roller coaster

and not drowning. You don't just sink and sink; there are ups after the downs.

Another feeling that seems alarming but is in fact normal in a startup is the feeling that what you're doing isn't working. The reason you can expect to feel this is that what you do probably won't work. Startups almost never get it right the first time. Much more commonly you launch something, and no one cares. Don't assume when this happens that you've failed. That's normal for startups. But don't sit around doing nothing. Iterate.

I like Paul Buchheit's suggestion of trying to make something that at least someone really loves. As long as you've made something that a few users are ecstatic about, you're on the right track. It will be good for your morale to have even a handful of users who really love you, and startups run on morale. But also it will tell you what to focus on. What is it about you that they love? Can you do more of that? Where can you find more people who love that sort of thing? As long as you have some core of users who love you, all you have to do is expand it. It may take a while, but as long as you keep plugging away, you'll win in the end. Both Blogger and Delicious did that. Both took years to succeed. But both began with a core of fanatically devoted users, and all Evan and Joshua had to do was grow that core incrementally. Wufoo is on the same trajectory now.

So when you release something and it seems like no one cares, look more closely. Are there zero users who really love you, or is there at least some little group that does? It's quite possible there will be zero. In that case, tweak your product and try again. Every one of you is working on a space that contains at least one winning permutation somewhere in it. If you just keep trying, you'll find it.

Let me mention some things not to do. The number one thing not to do is other things. If you find yourself saying a sentence that ends with "but we're going to keep working on the startup," you are in big trouble. Bob's going to grad school, but we're going to keep working on the startup. We're moving back to Minnesota, but we're going to keep working on the startup. We're taking on some consulting projects, but we're going to keep working on the startup. You may as well just translate these to "we're giving up on the startup, but we're not willing to admit that to ourselves," because that's what it means most of the time. A startup is so hard that working on it can't be preceded by "but."

In particular, don't go to graduate school, and don't start other projects. Distraction is fatal to startups. Going to (or back to) school is a huge predictor of death because in addition to the distraction it gives you something to say you're doing. If you're only doing a startup, then if the startup fails, you fail. If you're in grad school and your startup fails, you can say later "Oh yeah, we had this startup on the side when I was in grad school, but it didn't go anywhere."

You can't use euphemisms like "didn't go anywhere" for something that's your only occupation. People won't let you.

One of the most interesting things we've discovered from working on Y Combinator

is that founders are more motivated by the fear of looking bad than by the hope of getting millions of dollars. So if you want to get millions of dollars, put yourself in a position where failure will be public and humiliating.

When we first met the founders of Octopart, they seemed very smart, but not a great bet to succeed, because they didn't seem especially committed. One of the two founders was still in grad school. It was the usual story: he'd drop out if it looked like the startup was taking off. Since then he has not only dropped out of grad school, but appeared full length in Newsweek with the word "Billionaire" printed across his chest. He just cannot fail now. Everyone he knows has seen that picture. Girls who dissed him in high school have seen it. His mom probably has it on the fridge. It would be unthinkably humiliating to fail now. At this point he is committed to fight to the death.

I wish every startup we funded could appear in a Newsweek article describing them as the next generation of billionaires, because then none of them would be able to give up. The success rate would be 90%. I'm not kidding.

When we first knew the Octoparts they were lighthearted, cheery guys. Now when we talk to them they seem grimly determined. The electronic parts distributors are trying to squash them to keep their monopoly pricing. (If it strikes you as odd that people still order electronic parts out of thick paper catalogs in 2007, there's a reason for that. The distributors want to prevent the transparency that comes from having prices online.) I feel kind of bad that we've transformed these guys from lighthearted to grimly determined. But that comes with the territory. If a startup succeeds, you get millions of dollars, and you don't get that kind of money just by asking for it. You have to assume it takes some amount of pain.

And however tough things get for the Octoparts, I predict they'll succeed. They may have to morph themselves into something totally different, but they won't just crawl off and die. They're smart; they're working in a promising field; and they just cannot give up.

All of you guys already have the first two. You're all smart and working on promising ideas. Whether you end up among the living or the dead comes down to the third ingredient, not giving up.

So I'll tell you now: bad shit is coming. It always is in a startup. The odds of getting from launch to liquidity without some kind of disaster happening are one in a thousand. So don't get demoralized. When the disaster strikes, just say to yourself, ok, this was what Paul was talking about. What did he say to do? Oh, yeah. Don't give up.

[Japanese Translation]

[Arabic Translation]

News from the Front

September 2007

A few weeks ago I had a thought so heretical that it really surprised me. It may not matter all that much where you go to college.

For me, as for a lot of middle class kids, getting into a good college was more or less the meaning of life when I was growing up. What was I? A student. To do that well meant to get good grades. Why did one have to get good grades? To get into a good college. And why did one want to do that? There seemed to be several reasons: you'd learn more, get better jobs, make more money. But it didn't matter exactly what the benefits would be. College was a bottleneck through which all your future prospects passed; everything would be better if you went to a better college.

A few weeks ago I realized that somewhere along the line I had stopped believing that.

What first set me thinking about this was the new trend of worrying obsessively about what kindergarten your kids go to. It seemed to me this couldn't possibly matter. Either it won't help your kid get into Harvard, or if it does, getting into Harvard won't mean much anymore. And then I thought: how much does it mean even now?

It turns out I have a lot of data about that. My three partners and I run a seed stage investment firm called Y Combinator. We invest when the company is just a couple guys and an idea. The idea doesn't matter much; it will change anyway. Most of our decision is based on the founders. The average founder is three years out of college. Many have just graduated; a few are still in school. So we're in much the same position as a graduate program, or a company hiring people right out of college. Except our choices are immediately and visibly tested. There are two possible outcomes for a startup: success or failure—and usually you know within a year which it will be.

The test applied to a startup is among the purest of real world tests. A startup succeeds or fails depending almost entirely on the efforts of the founders. Success is decided by the market: you only succeed if users like what you've built. And users don't care where you went to college.

As well as having precisely measurable results, we have a lot of them. Instead of doing a small number of large deals like a traditional venture capital fund, we do a large number of small ones. We currently fund about 40 companies a year, selected from about 900 applications representing a total of about 2000 people. [1]

Between the volume of people we judge and the rapid, unequivocal test that's applied to our choices, Y Combinator has been an unprecedented opportunity for learning how to pick winners. One of the most surprising things we've learned is how little it matters where people went to college.

I thought I'd already been cured of caring about that. There's nothing like going to grad school at Harvard to cure you of any illusions you might have about the average Harvard undergrad. And yet Y Combinator showed us we were still overestimating people who'd been to elite colleges. We'd interview people from MIT or Harvard or Stanford and sometimes find ourselves thinking: they *must* be smarter than they seem. It took us a few iterations to learn to trust our senses.

Practically everyone thinks that someone who went to MIT or Harvard or Stanford must be smart. Even people who hate you for it believe it.

But when you think about what it means to have gone to an elite college, how could this be true? We're talking about a decision made by admissions officers—basically, HR people—based on a cursory examination of a huge pile of depressingly similar applications submitted by seventeen year olds. And what do they have to go on? An easily gamed standardized test; a short essay telling you what the kid thinks you want to hear; an interview with a random alum; a high school record that's largely an index of obedience. Who would rely on such a test?

And yet a lot of companies do. A lot of companies are very much influenced by where applicants went to college. How could they be? I think I know the answer to that.

There used to be a saying in the corporate world: "No one ever got fired for buying IBM." You no longer hear this about IBM specifically, but the idea is very much alive; there is a whole category of "enterprise" software companies that exist to take advantage of it. People buying technology for large organizations don't care if they pay a fortune for mediocre software. It's not their money. They just want to buy from a supplier who seems safe—a company with an established name, confident salesmen, impressive offices, and software that conforms to all the current fashions. Not necessarily a company that will deliver so much as one that, if they do let you down, will still seem to have been a prudent choice. So companies have evolved to fill that niche.

A recruiter at a big company is in much the same position as someone buying technology for one. If someone went to Stanford and is not obviously insane, they're probably a safe bet. And a safe bet is enough. No one ever measures recruiters by the later performance of people they turn down. [2]

I'm not saying, of course, that elite colleges have evolved to prey upon the weaknesses of large organizations the way enterprise software companies have. But they work as if they had. In addition to the power of the brand name, graduates of elite colleges have two critical qualities that plug right into the way large organizations work. They're good at doing what they're asked, since that's what it takes to please the adults who judge you at seventeen. And having been to an elite college makes them more confident.

Back in the days when people might spend their whole career at one big company, these qualities must have been very valuable. Graduates of elite colleges would have been capable, yet amenable to authority. And since individual performance is so hard to measure in large organizations, their own confidence would have been the starting point for their reputation.

Things are very different in the new world of startups. We couldn't save someone from the market's judgement even if we wanted to. And being charming and confident counts for nothing with users. All users care about is whether you make something they like. If you don't, you're dead.

Knowing that test is coming makes us work a lot harder to get the right answers than anyone would if they were merely hiring people. We can't afford to have any illusions about the predictors of success. And what we've found is that the variation between schools is so much smaller than the variation between individuals that it's negligible by comparison. We can learn more about someone in the first minute of talking to them than by knowing where they went to school.

It seems obvious when you put it that way. Look at the individual, not where they went to college. But that's a weaker statement than the idea I began with, that it doesn't matter much where a given individual goes to college. Don't you learn things at the best schools that you wouldn't learn at lesser places?

Apparently not. Obviously you can't prove this in the case of a single individual, but you can tell from aggregate evidence: you can't, without asking them, distinguish people who went to one school from those who went to another three times as far down the *US News* list. [3] Try it and see.

How can this be? Because how much you learn in college depends a lot more on you than the college. A determined party animal can get through the best school without learning anything. And someone with a real thirst for knowledge will be able to find a few smart people to learn from at a school that isn't prestigious at all.

The other students are the biggest advantage of going to an elite college; you learn more from them than the professors. But you should be able to reproduce this at most colleges if you make a conscious effort to find smart friends. At most colleges you can find at least a handful of other smart students, and most people have only a handful of close friends in college anyway. [4] The odds of finding

smart professors are even better. The curve for faculty is a lot flatter than for students, especially in math and the hard sciences; you have to go pretty far down the list of colleges before you stop finding smart professors in the math department.

So it's not surprising that we've found the relative prestige of different colleges useless in judging individuals. There's a lot of randomness in how colleges select people, and what they learn there depends much more on them than the college. Between these two sources of variation, the college someone went to doesn't mean a lot. It is to some degree a predictor of ability, but so weak that we regard it mainly as a source of error and try consciously to ignore it.

I doubt what we've discovered is an anomaly specific to startups. Probably people have always overestimated the importance of where one goes to college. We're just finally able to measure it.

The unfortunate thing is not just that people are judged by such a superficial test, but that so many judge themselves by it. A lot of people, probably the majority of people in America, have some amount of insecurity about where, or whether, they went to college. The tragedy of the situation is that by far the greatest liability of not having gone to the college you'd have liked is your own feeling that you're thereby lacking something. Colleges are a bit like exclusive clubs in this respect. There is only one real advantage to being a member of most exclusive clubs: you know you wouldn't be missing much if you weren't. When you're excluded, you can only imagine the advantages of being an insider. But invariably they're larger in your imagination than in real life.

So it is with colleges. Colleges differ, but they're nothing like the stamp of destiny so many imagine them to be. People aren't what some admissions officer decides about them at seventeen. They're what they make themselves.

Indeed, the great advantage of not caring where people went to college is not just that you can stop judging them (and yourself) by superficial measures, but that you can focus instead on what really matters. What matters is what you make of yourself. I think that's what we should tell kids. Their job isn't to get good grades so they can get into a good college, but to learn and do. And not just because that's more rewarding than worldly success. That will increasingly *be* the route to worldly success.

Notes

[1] Is what we measure worth measuring? I think so. You can get rich simply by being energetic and unscrupulous, but getting rich from a technology startup takes some amount of brains. It is just the kind of work the upper middle class values; it

has about the same intellectual component as being a doctor.

[2] Actually, someone did, once. Mitch Kapor's wife Freada was in charge of HR at Lotus in the early years. (As he is at pains to point out, they did not become romantically involved till afterward.) At one point they worried Lotus was losing its startup edge and turning into a big company. So as an experiment she sent their recruiters the resumes of the first 40 employees, with identifying details changed. These were the people who had made Lotus into the star it was. Not one got an interview.

[3] The *US News* list? Surely no one trusts that. Even if the statistics they consider are useful, how do they decide on the relative weights? The reason the *US News* list is meaningful is precisely because they are so intellectually dishonest in that respect. There is no external source they can use to calibrate the weighting of the statistics they use; if there were, we could just use that instead. What they must do is adjust the weights till the top schools are the usual suspects in about the right order. So in effect what the *US News* list tells us is what the editors think the top schools are, which is probably not far from the conventional wisdom on the matter. The amusing thing is, because some schools work hard to game the system, the editors will have to keep tweaking their algorithm to get the rankings they want.

[4] Possible doesn't mean easy, of course. A smart student at a party school will inevitably be something of an outcast, just as he or she would be in most high schools.

Thanks to Trevor Blackwell, Sarah Harlin, Jessica Livingston, Jackie McDonough, Peter Norvig, and Robert Morris for reading drafts of this.

French Translation

How to Do Philosophy

September 2007

In high school I decided I was going to study philosophy in college. I had several motives, some more honorable than others. One of the less honorable was to shock people. College was regarded as job training where I grew up, so studying philosophy seemed an impressively impractical thing to do. Sort of like slashing holes in your clothes or putting a safety pin through your ear, which were other forms of impressive impracticality then just coming into fashion.

But I had some more honest motives as well. I thought studying philosophy would be a shortcut straight to wisdom. All the people majoring in other things would just end up with a bunch of domain knowledge. I would be learning what was really what.

I'd tried to read a few philosophy books. Not recent ones; you wouldn't find those in our high school library. But I tried to read Plato and Aristotle. I doubt I believed I understood them, but they sounded like they were talking about something important. I assumed I'd learn what in college.

The summer before senior year I took some college classes. I learned a lot in the calculus class, but I didn't learn much in Philosophy 101. And yet my plan to study philosophy remained intact. It was my fault I hadn't learned anything. I hadn't read the books we were assigned carefully enough. I'd give Berkeley's *Principles of Human Knowledge* another shot in college. Anything so admired and so difficult to read must have something in it, if one could only figure out what.

Twenty-six years later, I still don't understand Berkeley. I have a nice edition of his collected works. Will I ever read it? Seems unlikely.

The difference between then and now is that now I understand why Berkeley is probably not worth trying to understand. I think I see now what went wrong with philosophy, and how we might fix it.

Words

I did end up being a philosophy major for most of college. It didn't work out as I'd hoped. I didn't learn any magical truths compared to which everything else was mere domain knowledge. But I do at least know now why I didn't. Philosophy doesn't really have a subject matter in the way math or history or most other university subjects do. There is no core of knowledge one must master. The closest you come to that is a knowledge of what various individual philosophers have said about different topics over the years. Few were sufficiently correct that people have forgotten who discovered what they discovered.

Formal logic has some subject matter. I took several classes in logic. I don't know if I learned anything from them. [1] It does seem to me very important to be able to flip ideas around in one's head: to see when two ideas don't fully cover the space of possibilities, or when one idea is the same as another but with a couple things changed. But did studying logic teach me the importance of thinking this way, or make me any better at it? I don't know.

There are things I know I learned from studying philosophy. The most dramatic I learned immediately, in the first semester of freshman year, in a class taught by Sydney Shoemaker. I learned that I don't exist. I am (and you are) a collection of cells that lurches around driven by various forces, and calls itself *I*. But there's no central, indivisible thing that your identity goes with. You could conceivably lose half your brain and live. Which means your brain could conceivably be split into two halves and each transplanted into different bodies. Imagine waking up after such an operation. You have to imagine being two people.

The real lesson here is that the concepts we use in everyday life are fuzzy, and break down if pushed too hard. Even a concept as dear to us as *I*. It took me a while to grasp this, but when I did it was fairly sudden, like someone in the nineteenth century grasping evolution and realizing the story of creation they'd been told as a child was all wrong. [2] Outside of math there's a limit to how far you can push words; in fact, it would not be a bad definition of math to call it the study of terms that have precise meanings. Everyday words are inherently imprecise. They work well enough in everyday life that you don't notice. Words seem to work, just as Newtonian physics seems to. But you can always make them break if you push them far enough.

I would say that this has been, unfortunately for philosophy, the central fact of philosophy. Most philosophical debates are not merely afflicted by but driven by confusions over words. Do we have free will? Depends what you mean by "free." Do abstract ideas exist? Depends what you mean by "exist."

Wittgenstein is popularly credited with the idea that most philosophical controversies are due to confusions over language. I'm not sure how much credit to give him. I suspect a lot of people realized this, but reacted simply by not studying philosophy, rather than becoming philosophy professors.

How did things get this way? Can something people have spent thousands of years studying really be a waste of time? Those are interesting questions. In fact, some of the most interesting questions you can ask about philosophy. The most valuable way to approach the current philosophical tradition may be neither to get lost in pointless speculations like Berkeley, nor to shut them down like Wittgenstein, but to study it as an example of reason gone wrong.

History

Western philosophy really begins with Socrates, Plato, and Aristotle. What we know of their predecessors comes from fragments and references in later works; their doctrines could be described as speculative cosmology that occasionally strays into analysis. Presumably they were driven by whatever makes people in every other society invent cosmologies. [3]

With Socrates, Plato, and particularly Aristotle, this tradition turned a corner. There started to be a lot more analysis. I suspect Plato and Aristotle were encouraged in this by progress in math. Mathematicians had by then shown that you could figure things out in a much more conclusive way than by making up fine sounding stories about them. [4]

People talk so much about abstractions now that we don't realize what a leap it must have been when they first started to. It was presumably many thousands of years between when people first started describing things as hot or cold and when someone asked "what is heat?" No doubt it was a very gradual process. We don't know if Plato or Aristotle were the first to ask any of the questions they did. But their works are the oldest we have that do this on a large scale, and there is a freshness (not to say naivete) about them that suggests some of the questions they asked were new to them, at least.

Aristotle in particular reminds me of the phenomenon that happens when people discover something new, and are so excited by it that they race through a huge percentage of the newly discovered territory in one lifetime. If so, that's evidence of how new this kind of thinking was. [5]

This is all to explain how Plato and Aristotle can be very impressive and yet naive and mistaken. It was impressive even to ask the questions they did. That doesn't mean they always came up with good answers. It's not considered insulting to say that ancient Greek mathematicians were naive in some respects, or at least lacked some concepts that would have made their lives easier. So I hope people will not be too offended if I propose that ancient philosophers were similarly naive. In particular, they don't seem to have fully grasped what I earlier called the central fact of philosophy: that words break if you push them too far.

"Much to the surprise of the builders of the first digital computers," Rod Brooks wrote, "programs written for them usually did not work." [6] Something similar happened when people first started trying to talk about abstractions. Much to their surprise, they didn't arrive at answers they agreed upon. In fact, they rarely

seemed to arrive at answers at all.

They were in effect arguing about artifacts induced by sampling at too low a resolution.

The proof of how useless some of their answers turned out to be is how little effect they have. No one after reading Aristotle's *Metaphysics* does anything differently as a result. [7]

Surely I'm not claiming that ideas have to have practical applications to be interesting? No, they may not have to. Hardy's boast that number theory had no use whatsoever wouldn't disqualify it. But he turned out to be mistaken. In fact, it's suspiciously hard to find a field of math that truly has no practical use. And Aristotle's explanation of the ultimate goal of philosophy in Book A of the *Metaphysics* implies that philosophy should be useful too.

Theoretical Knowledge

Aristotle's goal was to find the most general of general principles. The examples he gives are convincing: an ordinary worker builds things a certain way out of habit; a master craftsman can do more because he grasps the underlying principles. The trend is clear: the more general the knowledge, the more admirable it is. But then he makes a mistake—possibly the most important mistake in the history of philosophy. He has noticed that theoretical knowledge is often acquired for its own sake, out of curiosity, rather than for any practical need. So he proposes there are two kinds of theoretical knowledge: some that's useful in practical matters and some that isn't. Since people interested in the latter are interested in it for its own sake, it must be more noble. So he sets as his goal in the *Metaphysics* the exploration of knowledge that has no practical use. Which means no alarms go off when he takes on grand but vaguely understood questions and ends up getting lost in a sea of words.

His mistake was to confuse motive and result. Certainly, people who want a deep understanding of something are often driven by curiosity rather than any practical need. But that doesn't mean what they end up learning is useless. It's very valuable in practice to have a deep understanding of what you're doing; even if you're never called on to solve advanced problems, you can see shortcuts in the solution of simple ones, and your knowledge won't break down in edge cases, as it would if you were relying on formulas you didn't understand. Knowledge is power. That's what makes theoretical knowledge prestigious. It's also what causes smart people to be curious about certain things and not others; our DNA is not so disinterested as we might think.

So while ideas don't have to have immediate practical applications to be interesting, the kinds of things we find interesting will surprisingly often turn out to have practical applications.

The reason Aristotle didn't get anywhere in the *Metaphysics* was partly that he set

off with contradictory aims: to explore the most abstract ideas, guided by the assumption that they were useless. He was like an explorer looking for a territory to the north of him, starting with the assumption that it was located to the south.

And since his work became the map used by generations of future explorers, he sent them off in the wrong direction as well. [8] Perhaps worst of all, he protected them from both the criticism of outsiders and the promptings of their own inner compass by establishing the principle that the most noble sort of theoretical knowledge had to be useless.

The *Metaphysics* is mostly a failed experiment. A few ideas from it turned out to be worth keeping; the bulk of it has had no effect at all. The *Metaphysics* is among the least read of all famous books. It's not hard to understand the way Newton's *Principia* is, but the way a garbled message is.

Arguably it's an interesting failed experiment. But unfortunately that was not the conclusion Aristotle's successors derived from works like the *Metaphysics*. [9] Soon after, the western world fell on intellectual hard times. Instead of version 1s to be superseded, the works of Plato and Aristotle became revered texts to be mastered and discussed. And so things remained for a shockingly long time. It was not till around 1600 (in Europe, where the center of gravity had shifted by then) that one found people confident enough to treat Aristotle's work as a catalog of mistakes. And even then they rarely said so outright.

If it seems surprising that the gap was so long, consider how little progress there was in math between Hellenistic times and the Renaissance.

In the intervening years an unfortunate idea took hold: that it was not only acceptable to produce works like the *Metaphysics*, but that it was a particularly prestigious line of work, done by a class of people called philosophers. No one thought to go back and debug Aristotle's motivating argument. And so instead of correcting the problem Aristotle discovered by falling into it—that you can easily get lost if you talk too loosely about very abstract ideas—they continued to fall into it.

The Singularity

Curiously, however, the works they produced continued to attract new readers. Traditional philosophy occupies a kind of singularity in this respect. If you write in an unclear way about big ideas, you produce something that seems tantalizingly attractive to inexperienced but intellectually ambitious students. Till one knows better, it's hard to distinguish something that's hard to understand because the writer was unclear in his own mind from something like a mathematical proof that's hard to understand because the ideas it represents are hard to understand. To someone who hasn't learned the difference, traditional philosophy seems extremely attractive: as hard (and therefore impressive) as math, yet broader in scope. That was what lured me in as a high school student.

This singularity is even more singular in having its own defense built in. When things are hard to understand, people who suspect they're nonsense generally keep quiet. There's no way to prove a text is meaningless. The closest you can get is to show that the official judges of some class of texts can't distinguish them from placebos. [10]

And so instead of denouncing philosophy, most people who suspected it was a waste of time just studied other things. That alone is fairly damning evidence, considering philosophy's claims. It's supposed to be about the ultimate truths. Surely all smart people would be interested in it, if it delivered on that promise.

Because philosophy's flaws turned away the sort of people who might have corrected them, they tended to be self-perpetuating. Bertrand Russell wrote in a letter in 1912:

> Hitherto the people attracted to philosophy have been mostly those who loved the big generalizations, which were all wrong, so that few people with exact minds have taken up the subject. [11]

His response was to launch Wittgenstein at it, with dramatic results.

I think Wittgenstein deserves to be famous not for the discovery that most previous philosophy was a waste of time, which judging from the circumstantial evidence must have been made by every smart person who studied a little philosophy and declined to pursue it further, but for how he acted in response. [12] Instead of quietly switching to another field, he made a fuss, from inside. He was Gorbachev.

The field of philosophy is still shaken from the fright Wittgenstein gave it. [13] Later in life he spent a lot of time talking about how words worked. Since that seems to be allowed, that's what a lot of philosophers do now. Meanwhile, sensing a vacuum in the metaphysical speculation department, the people who used to do literary criticism have been edging Kantward, under new names like "literary theory," "critical theory," and when they're feeling ambitious, plain "theory." The writing is the familiar word salad:

> Gender is not like some of the other grammatical modes which express precisely a mode of conception without any reality that corresponds to the conceptual mode, and consequently do not express precisely something in reality by which the intellect could be moved to conceive a thing the way it does, even where that motive is not something in the thing as such. [14]

The singularity I've described is not going away. There's a market for writing that sounds impressive and can't be disproven. There will always be both supply and demand. So if one group abandons this territory, there will always be others ready to occupy it.

A Proposal

We may be able to do better. Here's an intriguing possibility. Perhaps we should do what Aristotle meant to do, instead of what he did. The goal he announces in the *Metaphysics* seems one worth pursuing: to discover the most general truths. That sounds good. But instead of trying to discover them because they're useless, let's try to discover them because they're useful.

I propose we try again, but that we use that heretofore despised criterion, applicability, as a guide to keep us from wondering off into a swamp of abstractions. Instead of trying to answer the question:

> What are the most general truths?

let's try to answer the question

> Of all the useful things we can say, which are the most general?

The test of utility I propose is whether we cause people who read what we've written to do anything differently afterward. Knowing we have to give definite (if implicit) advice will keep us from straying beyond the resolution of the words we're using.

The goal is the same as Aristotle's; we just approach it from a different direction.

As an example of a useful, general idea, consider that of the controlled experiment. There's an idea that has turned out to be widely applicable. Some might say it's part of science, but it's not part of any specific science; it's literally meta-physics (in our sense of "meta"). The idea of evolution is another. It turns out to have quite broad applications—for example, in genetic algorithms and even product design. Frankfurt's distinction between lying and bullshitting seems a promising recent example. [15]

These seem to me what philosophy should look like: quite general observations that would cause someone who understood them to do something differently.

Such observations will necessarily be about things that are imprecisely defined. Once you start using words with precise meanings, you're doing math. So starting from utility won't entirely solve the problem I described above—it won't flush out the metaphysical singularity. But it should help. It gives people with good intentions a new roadmap into abstraction. And they may thereby produce things that make the writing of the people with bad intentions look bad by comparison.

One drawback of this approach is that it won't produce the sort of writing that gets you tenure. And not just because it's not currently the fashion. In order to get tenure in any field you must not arrive at conclusions that members of tenure committees can disagree with. In practice there are two kinds of solutions to this problem. In math and the sciences, you can prove what you're saying, or at any rate adjust your conclusions so you're not claiming anything false ("6 of 8 subjects had lower blood pressure after the treatment"). In the humanities you can either avoid drawing any definite conclusions (e.g. conclude that an issue is a complex

one), or draw conclusions so narrow that no one cares enough to disagree with you.

The kind of philosophy I'm advocating won't be able to take either of these routes. At best you'll be able to achieve the essayist's standard of proof, not the mathematician's or the experimentalist's. And yet you won't be able to meet the usefulness test without implying definite and fairly broadly applicable conclusions. Worse still, the usefulness test will tend to produce results that annoy people: there's no use in telling people things they already believe, and people are often upset to be told things they don't.

Here's the exciting thing, though. Anyone can do this. Getting to general plus useful by starting with useful and cranking up the generality may be unsuitable for junior professors trying to get tenure, but it's better for everyone else, including professors who already have it. This side of the mountain is a nice gradual slope. You can start by writing things that are useful but very specific, and then gradually make them more general. Joe's has good burritos. What makes a good burrito? What makes good food? What makes anything good? You can take as long as you want. You don't have to get all the way to the top of the mountain. You don't have to tell anyone you're doing philosophy.

If it seems like a daunting task to do philosophy, here's an encouraging thought. The field is a lot younger than it seems. Though the first philosophers in the western tradition lived about 2500 years ago, it would be misleading to say the field is 2500 years old, because for most of that time the leading practitioners weren't doing much more than writing commentaries on Plato or Aristotle while watching over their shoulders for the next invading army. In the times when they weren't, philosophy was hopelessly intermingled with religion. It didn't shake itself free till a couple hundred years ago, and even then was afflicted by the structural problems I've described above. If I say this, some will say it's a ridiculously overbroad and uncharitable generalization, and others will say it's old news, but here goes: judging from their works, most philosophers up to the present have been wasting their time. So in a sense the field is still at the first step. [16]

That sounds a preposterous claim to make. It won't seem so preposterous in 10,000 years. Civilization always seems old, because it's always the oldest it's ever been. The only way to say whether something is really old or not is by looking at structural evidence, and structurally philosophy is young; it's still reeling from the unexpected breakdown of words.

Philosophy is as young now as math was in 1500. There is a lot more to discover.

Notes

[1] In practice formal logic is not much use, because despite some progress in the last 150 years we're still only able to formalize a small percentage of statements. We may never do that much better, for the same reason 1980s-style "knowledge representation" could never have worked; many statements may have no representation more concise than a huge, analog brain state.

[2] It was harder for Darwin's contemporaries to grasp this than we can easily imagine. The story of creation in the Bible is not just a Judeo-Christian concept; it's roughly what everyone must have believed since before people were people. The hard part of grasping evolution was to realize that species weren't, as they seem to be, unchanging, but had instead evolved from different, simpler organisms over unimaginably long periods of time.

Now we don't have to make that leap. No one in an industrialized country encounters the idea of evolution for the first time as an adult. Everyone's taught about it as a child, either as truth or heresy.

[3] Greek philosophers before Plato wrote in verse. This must have affected what they said. If you try to write about the nature of the world in verse, it inevitably turns into incantation. Prose lets you be more precise, and more tentative.

[4] Philosophy is like math's ne'er-do-well brother. It was born when Plato and Aristotle looked at the works of their predecessors and said in effect "why can't you be more like your brother?" Russell was still saying the same thing 2300 years later.

Math is the precise half of the most abstract ideas, and philosophy the imprecise half. It's probably inevitable that philosophy will suffer by comparison, because there's no lower bound to its precision. Bad math is merely boring, whereas bad philosophy is nonsense. And yet there are *some* good ideas in the imprecise half.

[5] Aristotle's best work was in logic and zoology, both of which he can be said to have invented. But the most dramatic departure from his predecessors was a new, much more analytical style of thinking. He was arguably the first scientist.

[6] Brooks, Rodney, *Programming in Common Lisp*, Wiley, 1985, p. 94.

[7] Some would say we depend on Aristotle more than we realize, because his ideas were one of the ingredients in our common culture. Certainly a lot of the words we use have a connection with Aristotle, but it seems a bit much to suggest that we wouldn't have the concept of the essence of something or the distinction between matter and form if Aristotle hadn't written about them.

One way to see how much we really depend on Aristotle would be to diff European culture with Chinese: what ideas did European culture have in 1800 that Chinese culture didn't, in virtue of Aristotle's contribution?

[8] The meaning of the word "philosophy" has changed over time. In ancient times it covered a broad range of topics, comparable in scope to our "scholarship" (though without the methodological implications). Even as late as Newton's time it included what we now call "science." But core of the subject today is still what seemed to Aristotle the core: the attempt to discover the most general truths.

Aristotle didn't call this "metaphysics." That name got assigned to it because the books we now call the *Metaphysics* came after (meta = after) the *Physics* in the standard edition of Aristotle's works compiled by Andronicus of Rhodes three centuries later. What we call "metaphysics" Aristotle called "first philosophy."

[9] Some of Aristotle's immediate successors may have realized this, but it's hard to say because most of their works are lost.

[10] Sokal, Alan, "Transgressing the Boundaries: Toward a Transformative Hermeneutics of Quantum Gravity," *Social Text* 46/47, pp. 217-252.

Abstract-sounding nonsense seems to be most attractive when it's aligned with some axe the audience already has to grind. If this is so we should find it's most popular with groups that are (or feel) weak. The powerful don't need its reassurance.

[11] Letter to Ottoline Morrell, December 1912. Quoted in:

Monk, Ray, *Ludwig Wittgenstein: The Duty of Genius*, Penguin, 1991, p. 75.

[12] A preliminary result, that all metaphysics between Aristotle and 1783 had been a waste of time, is due to I. Kant.

[13] Wittgenstein asserted a sort of mastery to which the inhabitants of early 20th century Cambridge seem to have been peculiarly vulnerable—perhaps partly because so many had been raised religious and then stopped believing, so had a vacant space in their heads for someone to tell them what to do (others chose Marx or Cardinal Newman), and partly because a quiet, earnest place like Cambridge in that era had no natural immunity to messianic figures, just as European politics then had no natural immunity to dictators.

[14] This is actually from the *Ordinatio* of Duns Scotus (ca. 1300), with "number" replaced by "gender." Plus ca change.

Wolter, Allan (trans), *Duns Scotus: Philosophical Writings*, Nelson, 1963, p. 92.

[15] Frankfurt, Harry, *On Bullshit*, Princeton University Press, 2005.

[16] Some introductions to philosophy now take the line that philosophy is worth studying as a process rather than for any particular truths you'll learn. The philosophers whose works they cover would be rolling in their graves at that. They

hoped they were doing more than serving as examples of how to argue: they hoped they were getting results. Most were wrong, but it doesn't seem an impossible hope.

This argument seems to me like someone in 1500 looking at the lack of results achieved by alchemy and saying its value was as a process. No, they were going about it wrong. It turns out it is possible to transmute lead into gold (though not economically at current energy prices), but the route to that knowledge was to backtrack and try another approach.

Thanks to Trevor Blackwell, Paul Buchheit, Jessica Livingston, Robert Morris, Mark Nitzberg, and Peter Norvig for reading drafts of this.

French Translation

The Future of Web Startups

October 2007

(This essay is derived from a keynote at FOWA in October 2007.)

There's something interesting happening right now. Startups are undergoing the same transformation that technology does when it becomes cheaper.

It's a pattern we see over and over in technology. Initially there's some device that's very expensive and made in small quantities. Then someone discovers how to make them cheaply; many more get built; and as a result they can be used in new ways.

Computers are a familiar example. When I was a kid, computers were big, expensive machines built one at a time. Now they're a commodity. Now we can stick computers in everything.

This pattern is very old. Most of the turning points in economic history are instances of it. It happened to steel in the 1850s, and to power in the 1780s. It happened to cloth manufacture in the thirteenth century, generating the wealth that later brought about the Renaissance. Agriculture itself was an instance of this pattern.

Now as well as being produced by startups, this pattern is happening *to* startups. It's so cheap to start web startups that orders of magnitudes more will be started. If the pattern holds true, that should cause dramatic changes.

1. Lots of Startups

So my first prediction about the future of web startups is pretty straightforward: there will be a lot of them. When starting a startup was expensive, you had to get the permission of investors to do it. Now the only threshold is courage.

Even that threshold is getting lower, as people watch others take the plunge and survive. In the last batch of startups we funded, we had several founders who said they'd thought of applying before, but weren't sure and got jobs instead. It was only after hearing reports of friends who'd done it that they decided to try it themselves.

Starting a startup is hard, but having a 9 to 5 job is hard too, and in some ways a worse kind of hard. In a startup you have lots of worries, but you don't have that feeling that your life is flying by like you do in a big company. Plus in a startup you could make much more money.

As word spreads that startups work, the number may grow to a point that would now seem surprising.

We now think of it as normal to have a job at a company, but this is the thinnest of historical veneers. Just two or three lifetimes ago, most people in what are now called industrialized countries lived by farming. So while it may seem surprising to propose that large numbers of people will change the way they make a living, it would be more surprising if they didn't.

2. Standardization

When technology makes something dramatically cheaper, standardization always follows. When you make things in large volumes you tend to standardize everything that doesn't need to change.

At Y Combinator we still only have four people, so we try to standardize everything. We could hire employees, but we want to be forced to figure out how to scale investing.

We often tell startups to release a minimal version one quickly, then let the needs of the users determine what to do next. In essense, let the market design the product. We've done the same thing ourselves. We think of the techniques we're developing for dealing with large numbers of startups as like software. Sometimes it literally is software, like Hacker News and our application system.

One of the most important things we've been working on standardizing are investment terms. Till now investment terms have been individually negotiated. This is a problem for founders, because it makes raising money take longer and cost more in legal fees. So as well as using the same paperwork for every deal we do, we've commissioned generic angel paperwork that all the startups we fund can use for future rounds.

Some investors will still want to cook up their own deal terms. Series A rounds, where you raise a million dollars or more, will be custom deals for the forseeable future. But I think angel rounds will start to be done mostly with standardized agreements. An angel who wants to insert a bunch of complicated terms into the agreement is probably not one you want anyway.

3. New Attitude to Acquisition

Another thing I see starting to get standardized is acquisitions. As the volume of startups increases, big companies will start to develop standardized procedures

that make acquisitions little more work than hiring someone.

Google is the leader here, as in so many areas of technology. They buy a lot of startups— more than most people realize, because they only announce a fraction of them. And being Google, they're figuring out how to do it efficiently.

One problem they've solved is how to think about acquisitions. For most companies, acquisitions still carry some stigma of inadequacy. Companies do them because they have to, but there's usually some feeling they shouldn't have to— that their own programmers should be able to build everything they need.

Google's example should cure the rest of the world of this idea. Google has by far the best programmers of any public technology company. If they don't have a problem doing acquisitions, the others should have even less problem. However many Google does, Microsoft should do ten times as many.

One reason Google doesn't have a problem with acquisitions is that they know first-hand the quality of the people they can get that way. Larry and Sergey only started Google after making the rounds of the search engines trying to sell their idea and finding no takers. They've *been* the guys coming in to visit the big company, so they know who might be sitting across that conference table from them.

4. Riskier Strategies are Possible

Risk is always proportionate to reward. The way to get really big returns is to do things that seem crazy, like starting a new search engine in 1998, or turning down a billion dollar acquisition offer.

This has traditionally been a problem in venture funding. Founders and investors have different attitudes to risk. Knowing that risk is on average proportionate to reward, investors like risky strategies, while founders, who don't have a big enough sample size to care what's true on average, tend to be more conservative.

If startups are easy to start, this conflict goes away, because founders can start them younger, when it's rational to take more risk, and can start more startups total in their careers. When founders can do lots of startups, they can start to look at the world in the same portfolio-optimizing way as investors. And that means the overall amount of wealth created can be greater, because strategies can be riskier.

5. Younger, Nerdier Founders

If startups become a cheap commodity, more people will be able to have them, just as more people could have computers once microprocessors made them cheap. And in particular, younger and more technical founders will be able to start startups than could before.

Back when it cost a lot to start a startup, you had to convince investors to let you

do it. And that required very different skills from actually doing the startup. If investors were perfect judges, the two would require exactly the same skills. But unfortunately most investors are terrible judges. I know because I see behind the scenes what an enormous amount of work it takes to raise money, and the amount of selling required in an industry is always inversely proportional to the judgement of the buyers.

Fortunately, if startups get cheaper to start, there's another way to convince investors. Instead of going to venture capitalists with a business plan and trying to convince them to fund it, you can get a product launched on a few tens of thousands of dollars of seed money from us or your uncle, and approach them with a working company instead of a plan for one. Then instead of having to seem smooth and confident, you can just point them to Alexa.

This way of convincing investors is better suited to hackers, who often went into technology in part because they felt uncomfortable with the amount of fakeness required in other fields.

6. Startup Hubs Will Persist

It might seem that if startups get cheap to start, it will mean the end of startup hubs like Silicon Valley. If all you need to start a startup is rent money, you should be able to do it anywhere.

This is kind of true and kind of false. It's true that you can now *start* a startup anywhere. But you have to do more with a startup than just start it. You have to make it succeed. And that is more likely to happen in a startup hub.

I've thought a lot about this question, and it seems to me the increasing cheapness of web startups will if anything increase the importance of startup hubs. The value of startup hubs, like centers for any kind of business, lies in something very old-fashioned: face to face meetings. No technology in the immediate future will replace walking down University Ave and running into a friend who tells you how to fix a bug that's been bothering you all weekend, or visiting a friend's startup down the street and ending up in a conversation with one of their investors.

The question of whether to be in a startup hub is like the question of whether to take outside investment. The question is not whether you *need* it, but whether it brings any advantage at all. Because anything that brings an advantage will give your competitors an advantage over you if they do it and you don't. So if you hear someone saying "we don't need to be in Silicon Valley," that use of the word "need" is a sign they're not even thinking about the question right.

And while startup hubs are as powerful magnets as ever, the increasing cheapness of starting a startup means the particles they're attracting are getting lighter. A startup now can be just a pair of 22 year old guys. A company like that can move much more easily than one with 10 people, half of whom have kids.

We know because we make people move for Y Combinator, and it doesn't seem to be a problem. The advantage of being able to work together face to face for three months outweighs the inconvenience of moving. Ask anyone who's done it.

The mobility of seed-stage startups means that seed funding is a national business. One of the most common emails we get is from people asking if we can help them set up a local clone of Y Combinator. But this just wouldn't work. Seed funding isn't regional, just as big research universities aren't.

Is seed funding not merely national, but international? Interesting question. There are signs it may be. We've had an ongoing stream of founders from outside the US, and they tend to do particularly well, because they're all people who were so determined to succeed that they were willing to move to another country to do it.

The more mobile startups get, the harder it would be to start new silicon valleys. If startups are mobile, the best local talent will go to the real Silicon Valley, and all they'll get at the local one will be the people who didn't have the energy to move.

This is not a nationalistic idea, incidentally. It's cities that compete, not countries. Atlanta is just as hosed as Munich.

7. Better Judgement Needed

If the number of startups increases dramatically, then the people whose job is to judge them are going to have to get better at it. I'm thinking particularly of investors and acquirers. We now get on the order of 1000 applications a year. What are we going to do if we get 10,000?

That's actually an alarming idea. But we'll figure out some kind of answer. We'll have to. It will probably involve writing some software, but fortunately we can do that.

Acquirers will also have to get better at picking winners. They generally do better than investors, because they pick later, when there's more performance to measure. But even at the most advanced acquirers, identifying companies to buy is extremely ad hoc, and completing the acquisition often involves a great deal of unneccessary friction.

I think acquirers may eventually have chief acquisition officers who will both identify good acquisitions and make the deals happen. At the moment those two functions are separate. Promising new startups are often discovered by developers. If someone powerful enough wants to buy them, the deal is handed over to corp dev guys to negotiate. It would be better if both were combined in one group, headed by someone with a technical background and some vision of what they wanted to accomplish. Maybe in the future big companies will have both a VP of Engineering responsible for technology developed in-house, and a CAO responsible for bringing technology in from outside.

At the moment, there is no one within big companies who gets in trouble when they buy a startup for $200 million that they could have bought earlier for $20 million. There should start to be someone who gets in trouble for that.

8. College Will Change

If the best hackers start their own companies after college instead of getting jobs, that will change what happens in college. Most of these changes will be for the better. I think the experience of college is warped in a bad way by the expectation that afterward you'll be judged by potential employers.

One change will be in the meaning of "after college," which will switch from when one graduates from college to when one leaves it. If you're starting your own company, why do you need a degree? We don't encourage people to start startups during college, but the best founders are certainly capable of it. Some of the most successful companies we've funded were started by undergrads.

I grew up in a time where college degrees seemed really important, so I'm alarmed to be saying things like this, but there's nothing magical about a degree. There's nothing that magically changes after you take that last exam. The importance of degrees is due solely to the administrative needs of large organizations. These can certainly affect your life—it's hard to get into grad school, or to get a work visa in the US, without an undergraduate degree—but tests like this will matter less and less.

As well as mattering less whether students get degrees, it will also start to matter less where they go to college. In a startup you're judged by users, and they don't care where you went to college. So in a world of startups, elite universities will play less of a role as gatekeepers. In the US it's a national scandal how easily children of rich parents game college admissions. But the way this problem ultimately gets solved may not be by reforming the universities but by going around them. We in the technology world are used to that sort of solution: you don't beat the incumbents; you redefine the problem to make them irrelevant.

The greatest value of universities is not the brand name or perhaps even the classes so much as the people you meet. If it becomes common to start a startup after college, students may start trying to maximize this. Instead of focusing on getting internships at companies they want to work for, they may start to focus on working with other students they want as cofounders.

What students do in their classes will change too. Instead of trying to get good grades to impress future employers, students will try to learn things. We're talking about some pretty dramatic changes here.

9. Lots of Competitors

If it gets easier to start a startup, it's easier for competitors too. That doesn't erase

the advantage of increased cheapness, however. You're not all playing a zero-sum game. There's not some fixed number of startups that can succeed, regardless of how many are started.

In fact, I don't think there's any limit to the number of startups that could succeed. Startups succeed by creating wealth, which is the satisfaction of people's desires. And people's desires seem to be effectively infinite, at least in the short term.

What the increasing number of startups does mean is that you won't be able to sit on a good idea. Other people have your idea, and they'll be increasingly likely to do something about it.

10. Faster Advances

There's a good side to that, at least for consumers of technology. If people get right to work implementing ideas instead of sitting on them, technology will evolve faster.

Some kinds of innovations happen a company at a time, like the punctuated equilibrium model of evolution. There are some kinds of ideas that are so threatening that it's hard for big companies even to think of them. Look at what a hard time Microsoft is having discovering web apps. They're like a character in a movie that everyone in the audience can see something bad is about to happen to, but who can't see it himself. The big innovations that happen a company at a time will obviously happen faster if the rate of new companies increases.

But in fact there will be a double speed increase. People won't wait as long to act on new ideas, but also those ideas will increasingly be developed within startups rather than big companies. Which means technology will evolve faster per company as well.

Big companies are just not a good place to make things happen fast. I talked recently to a founder whose startup had been acquired by a big company. He was a precise sort of guy, so he'd measured their productivity before and after. He counted lines of code, which can be a dubious measure, but in this case was meaningful because it was the same group of programmers. He found they were one thirteenth as productive after the acquisition.

The company that bought them was not a particularly stupid one. I think what he was measuring was mostly the cost of bigness. I experienced this myself, and his number sounds about right. There's something about big companies that just sucks the energy out of you.

Imagine what all that energy could do if it were put to use. There is an enormous latent capacity in the world's hackers that most people don't even realize is there. That's the main reason we do Y Combinator: to let loose all this energy by making it easy for hackers to start their own startups.

A Series of Tubes

The process of starting startups is currently like the plumbing in an old house. The pipes are narrow and twisty, and there are leaks in every joint. In the future this mess will gradually be replaced by a single, huge pipe. The water will still have to get from A to B, but it will get there faster and without the risk of spraying out through some random leak.

This will change a lot of things for the better. In a big, straight pipe like that, the force of being measured by one's performance will propagate back through the whole system. Performance is always the ultimate test, but there are so many kinks in the plumbing now that most people are insulated from it most of the time. So you end up with a world in which high school students think they need to get good grades to get into elite colleges, and college students think they need to get good grades to impress employers, within which the employees waste most of their time in political battles, and from which consumers have to buy anyway because there are so few choices. Imagine if that sequence became a big, straight pipe. Then the effects of being measured by performance would propagate all the way back to high school, flushing out all the arbitrary stuff people are measured by now. That is the future of web startups.

Thanks to Brian Oberkirch and Simon Willison for inviting me to speak, and the crew at Carson Systems for making everything run smoothly.

- Japanese Translation

Why to Move to a Startup Hub

October 2007

After the last talk I gave, one of the organizers got up on the stage to deliver an impromptu rebuttal. That never happened before. I only heard the first few sentences, but that was enough to tell what I said that upset him: that startups would do better if they moved to Silicon Valley.

This conference was in London, and most of the audience seemed to be from the UK. So saying startups should move to Silicon Valley seemed like a nationalistic remark: an obnoxious American telling them that if they wanted to do things right they should all just move to America.

Actually I'm less American than I seem. I didn't say so, but I'm British by birth. And just as Jews are ex officio allowed to tell Jewish jokes, I don't feel like I have to bother being diplomatic with a British audience.

The idea that startups would do better to move to Silicon Valley is not even a nationalistic one. [1] It's the same thing I say to startups in the US. Y Combinator alternates between coasts every 6 months. Every other funding cycle is in Boston. And even though Boston is the second biggest startup hub in the US (and the world), we tell the startups from those cycles that their best bet is to move to Silicon Valley. If that's true of Boston, it's even more true of every other city.

This is about cities, not countries.

And I think I can prove I'm right. You can easily reduce the opposing argument ad what most people would agree was absurdum. Few would be willing to claim that it doesn't matter at all where a startup is—that a startup operating out of a small agricultural town wouldn't benefit from moving to a startup hub. Most people could see how it might be helpful to be in a place where there was infrastructure for

startups, accumulated knowledge about how to make them work, and other people trying to do it. And yet whatever argument you use to prove that startups don't need to move from London to Silicon Valley could equally well be used to prove startups don't need to move from smaller towns to London.

The difference between cities is a matter of degree. And if, as nearly everyone who knows agrees, startups are better off in Silicon Valley than Boston, then they're better off in Silicon Valley than everywhere else too.

I realize I might seem to have a vested interest in this conclusion, because startups that move to the US might do it through Y Combinator. But the American startups we've funded will attest that I say the same thing to them.

I'm not claiming of course that every startup has to go to Silicon Valley to succeed. Just that all other things being equal, the more of a startup hub a place is, the better startups will do there. But other considerations can outweigh the advantages of moving. I'm not saying founders with families should uproot them to move halfway around the world; that might be too much of a distraction.

Immigration difficulties might be another reason to stay put. Dealing with immigration problems is like raising money: for some reason it seems to consume all your attention. A startup can't afford much of that. One Canadian startup we funded spent about 6 months working on moving to the US. Eventually they just gave up, because they couldn't afford to take so much time away from working on their software.

(If another country wanted to establish a rival to Silicon Valley, the single best thing they could do might be to create a special visa for startup founders. US immigration policy is one of Silicon Valley's biggest weaknesses.)

If your startup is connected to a specific industry, you may be better off in one of its centers. A startup doing something related to entertainment might want to be in New York or LA.

And finally, if a good investor has committed to fund you if you stay where you are, you should probably stay. Finding investors is hard. You generally shouldn't pass up a definite funding offer to move. [2]

In fact, the quality of the investors may be the main advantage of startup hubs. Silicon Valley investors are noticeably more aggressive than Boston ones. Over and over, I've seen startups we've funded snatched by west coast investors out from under the noses of Boston investors who saw them first but acted too slowly. At this year's Boston Demo Day, I told the audience that this happened every year, so if they saw a startup they liked, they should make them an offer. And yet within a month it had happened again: an aggressive west coast VC who had met the founder of a YC-funded startup a week before beat out a Boston VC who had known him for years. By the time the Boston VC grasped what was happening, the deal was already gone.

Boston investors will admit they're more conservative. Some want to believe this comes from the city's prudent Yankee character. But Occam's razor suggests the truth is less flattering. Boston investors are probably more conservative than Silicon Valley investors for the same reason Chicago investors are more conservative than Boston ones. They don't understand startups as well.

West coast investors aren't bolder because they're irresponsible cowboys, or because the good weather makes them optimistic. They're bolder because they know what they're doing. They're the skiers who ski on the diamond slopes. Boldness is the essence of venture investing. The way you get big returns is not by trying to avoid losses, but by trying to ensure you get some of the big hits. And the big hits often look risky at first.

Like Facebook. Facebook was started in Boston. Boston VCs had the first shot at them. But they said no, so Facebook moved to Silicon Valley and raised money there. The partner who turned them down now says that "may turn out to have been a mistake."

Empirically, boldness wins. If the aggressive ways of west coast investors are going to come back to bite them, it has been a long time coming. Silicon Valley has been pulling ahead of Boston since the 1970s. If there was going to be a comeuppance for the west coast investors, the bursting of the Bubble would have been it. But since then the west coast has just pulled further ahead.

West coast investors are confident enough of their judgement to act boldly; east coast investors, not so much; but anyone who thinks east coast investors act that way out of prudence should see the frantic reactions of an east coast VC in the process of losing a deal to a west coast one.

In addition to the concentration that comes from specialization, startup hubs are also markets. And markets are usually centralized. Even now, when traders could be anywhere, they cluster in a few cities. It's hard to say exactly what it is about face to face contact that makes deals happen, but whatever it is, it hasn't yet been duplicated by technology.

Walk down University Ave at the right time, and you might overhear five different people talking on the phone about deals. In fact, this is part of the reason Y Combinator is in Boston half the time: it's hard to stand that year round. But though it can sometimes be annoying to be surrounded by people who only think about one thing, it's the place to be if that one thing is what you're trying to do.

I was talking recently to someone who works on search at Google. He knew a lot of people at Yahoo, so he was in a good position to compare the two companies. I asked him why Google was better at search. He said it wasn't anything specific Google did, but simply that they understood search so much better.

And that's why startups thrive in startup hubs like Silicon Valley. Startups are a

very specialized business, as specialized as diamond cutting. And in startup hubs they understand it.

Notes

[1] The nationalistic idea is the converse: that startups should stay in a certain city because of the country it's in. If you really have a "one world" viewpoint, deciding to move from London to Silicon Valley is no different from deciding to move from Chicago to Silicon Valley.

[2] An investor who merely seems like he will fund you, however, you can ignore. Seeming like they will fund you one day is the way investors say No.

Thanks to Sam Altman, Jessica Livingston, Harjeet Taggar, and Kulveer Taggar for reading drafts of this.

 Comment on this essay.

Japanese Translation

Six Principles for Making New Things

February 2008

The fiery reaction to the release of Arc had an unexpected consequence: it made me realize I had a design philosophy. The main complaint of the more articulate critics was that Arc seemed so flimsy. After years of working on it, all I had to show for myself were a few thousand lines of macros? Why hadn't I worked on more substantial problems?

As I was mulling over these remarks it struck me how familiar they seemed. This was exactly the kind of thing people said at first about Viaweb, and Y Combinator, and most of my essays.

When we launched Viaweb, it seemed laughable to VCs and e-commerce "experts." We were just a couple guys in an apartment, which did not seem cool in 1995 the way it does now. And the thing we'd built, as far as they could tell, wasn't even software. Software, to them, equalled big, honking Windows apps. Since Viaweb was the first web-based app they'd seen, it seemed to be nothing more than a website. They were even more contemptuous when they discovered that Viaweb didn't process credit card transactions (we didn't for the whole first year). Transaction processing seemed to them what e-commerce was all about. It sounded serious and difficult.

And yet, mysteriously, Viaweb ended up crushing all its competitors.

The initial reaction to Y Combinator was almost identical. It seemed laughably lightweight. Startup funding meant series A rounds: millions of dollars given to a small number of startups founded by people with established credentials after months of serious, businesslike meetings, on terms described in a document a foot thick. Y Combinator seemed inconsequential. It's too early to say yet whether Y Combinator will turn out like Viaweb, but judging from the number of imitations, a lot of people seem to think we're on to something.

I can't measure whether my essays are successful, except in page views, but the reaction to them is at least different from when I started. At first the default reaction of the Slashdot trolls was (translated into articulate terms): "Who is this guy and what authority does he have to write about these topics? I haven't read the essay, but there's no way anything so short and written in such an informal style could have anything useful to say about such and such topic, when people with degrees in the subject have already written many thick books about it." Now there's a new generation of trolls on a new generation of sites, but they have at least started to omit the initial "Who is this guy?"

Now people are saying the same things about Arc that they said at first about Viaweb and Y Combinator and most of my essays. Why the pattern? The answer, I realized, is that my m.o. for all four has been the same.

Here it is: I like to find (a) simple solutions (b) to overlooked problems (c) that actually need to be solved, and (d) deliver them as informally as possible, (e) starting with a very crude version 1, then (f) iterating rapidly.

When I first laid out these principles explicitly, I noticed something striking: this is practically a recipe for generating a contemptuous initial reaction. Though simple solutions are better, they don't seem as impressive as complex ones. Overlooked problems are by definition problems that most people think don't matter. Delivering solutions in an informal way means that instead of judging something by the way it's presented, people have to actually understand it, which is more work. And starting with a crude version 1 means your initial effort is always small and incomplete.

I'd noticed, of course, that people never seemed to grasp new ideas at first. I thought it was just because most people were stupid. Now I see there's more to it than that. Like a contrarian investment fund, someone following this strategy will almost always be doing things that seem wrong to the average person.

As with contrarian investment strategies, that's exactly the point. This technique is successful (in the long term) because it gives you all the advantages other people forgo by trying to seem legit. If you work on overlooked problems, you're more likely to discover new things, because you have less competition. If you deliver solutions informally, you (a) save all the effort you would have had to expend to make them look impressive, and (b) avoid the danger of fooling yourself as well as your audience. And if you release a crude version 1 then iterate, your solution can benefit from the imagination of nature, which, as Feynman pointed out, is more powerful than your own.

In the case of Viaweb, the simple solution was to make the software run on the server. The overlooked problem was to generate web sites automatically; in 1995, online stores were all made by hand by human designers, but we knew this wouldn't scale. The part that actually mattered was graphic design, not transaction processing. The informal delivery mechanism was me, showing up in jeans and a t-

shirt at some retailer's office. And the crude version 1 was, if I remember correctly, less than 10,000 lines of code when we launched.

The power of this technique extends beyond startups and programming languages and essays. It probably extends to any kind of creative work. Certainly it can be used in painting: this is exactly what Cezanne and Klee did.

At Y Combinator we bet money on it, in the sense that we encourage the startups we fund to work this way. There are always new ideas right under your nose. So look for simple things that other people have overlooked—things people will later claim were "obvious"—especially when they've been led astray by obsolete conventions, or by trying to do things that are superficially impressive. Figure out what the real problem is, and make sure you solve that. Don't worry about trying to look corporate; the product is what wins in the long term. And launch as soon as you can, so you start learning from users what you should have been making.

Reddit is a classic example of this approach. When Reddit first launched, it seemed like there was nothing to it. To the graphically unsophisticated its deliberately minimal design seemed like no design at all. But Reddit solved the real problem, which was to tell people what was new and otherwise stay out of the way. As a result it became massively successful. Now that conventional ideas have caught up with it, it seems obvious. People look at Reddit and think the founders were lucky. Like all such things, it was harder than it looked. The Reddits pushed so hard against the current that they reversed it; now it looks like they're merely floating downstream.

So when you look at something like Reddit and think "I wish I could think of an idea like that," remember: ideas like that are all around you. But you ignore them because they look wrong.

Trolls

February 2008

A user on Hacker News recently posted a comment that set me thinking:

> Something about hacker culture that never really set well with me was this — the nastiness. ... I just don't understand why people troll like they do.

I've thought a lot over the last couple years about the problem of trolls. It's an old one, as old as forums, but we're still just learning what the causes are and how to address them.

There are two senses of the word "troll." In the original sense it meant someone, usually an outsider, who deliberately stirred up fights in a forum by saying controversial things. [1] For example, someone who didn't use a certain programming language might go to a forum for users of that language and make disparaging remarks about it, then sit back and watch as people rose to the bait. This sort of trolling was in the nature of a practical joke, like letting a bat loose in a room full of people.

The definition then spread to people who behaved like assholes in forums, whether intentionally or not. Now when people talk about trolls they usually mean this broader sense of the word. Though in a sense this is historically inaccurate, it is in other ways more accurate, because when someone is being an asshole it's usually uncertain even in their own mind how much is deliberate. That is arguably one of the defining qualities of an asshole.

I think trolling in the broader sense has four causes. The most important is distance. People will say things in anonymous forums that they'd never dare say to someone's face, just as they'll do things in cars that they'd never do as pedestrians — like tailgate people, or honk at them, or cut them off.

Trolling tends to be particularly bad in forums related to computers, and I think that's due to the kind of people you find there. Most of them (myself included) are more comfortable dealing with abstract ideas than with people. Hackers can be abrupt even in person. Put them on an anonymous forum, and the problem gets worse.

The third cause of trolling is incompetence. If you disagree with something, it's

easier to say "you suck" than to figure out and explain exactly what you disagree with. You're also safe that way from refutation. In this respect trolling is a lot like graffiti. Graffiti happens at the intersection of ambition and incompetence: people want to make their mark on the world, but have no other way to do it than literally making a mark on the world. [2]

The final contributing factor is the culture of the forum. Trolls are like children (many *are* children) in that they're capable of a wide range of behavior depending on what they think will be tolerated. In a place where rudeness isn't tolerated, most can be polite. But vice versa as well.

There's a sort of Gresham's Law of trolls: trolls are willing to use a forum with a lot of thoughtful people in it, but thoughtful people aren't willing to use a forum with a lot of trolls in it. Which means that once trolling takes hold, it tends to become the dominant culture. That had already happened to Slashdot and Digg by the time I paid attention to comment threads there, but I watched it happen to Reddit.

News.YC is, among other things, an experiment to see if this fate can be avoided. The sites's guidelines explicitly ask people not to say things they wouldn't say face to face. If someone starts being rude, other users will step in and tell them to stop. And when people seem to be deliberately trolling, we ban them ruthlessly.

Technical tweaks may also help. On Reddit, votes on your comments don't affect your karma score, but they do on News.YC. And it does seem to influence people when they can see their reputation in the eyes of their peers drain away after making an asshole remark. Often users have second thoughts and delete such comments.

One might worry this would prevent people from expressing controversial ideas, but empirically that doesn't seem to be what happens. When people say something substantial that gets modded down, they stubbornly leave it up. What people delete are wisecracks, because they have less invested in them.

So far the experiment seems to be working. The level of conversation on News.YC is as high as on any forum I've seen. But we still only have about 8,000 uniques a day. The conversations on Reddit were good when it was that small. The challenge is whether we can keep things this way.

I'm optimistic we will. We're not depending just on technical tricks. The core users of News.YC are mostly refugees from other sites that were overrun by trolls. They feel about trolls roughly the way refugees from Cuba or Eastern Europe feel about dictatorships. So there are a lot of people working to keep this from happening again.

Notes

[1] I mean forum in the general sense of a place to exchange views. The original Internet forums were not web sites but Usenet newsgroups.

[2] I'm talking here about everyday tagging. Some graffiti is quite impressive (anything becomes art if you do it well enough) but the median tag is just visual spam.

- Russian Translation

A New Venture Animal

March 2008, rev May 2013

(This essay grew out of something I wrote for myself to figure out what we do. Even though Y Combinator is now 3 years old, we're still trying to understand its implications.)

I was annoyed recently to read a description of Y Combinator that said "Y Combinator does seed funding for startups." What was especially annoying about it was that I wrote it. This doesn't really convey what we do. And the reason it's inaccurate is that, paradoxically, funding very early stage startups is not mainly about funding.

Saying YC does seed funding for startups is a description in terms of earlier models. It's like calling a car a horseless carriage.

When you scale animals you can't just keep everything in proportion. For example, volume grows as the cube of linear dimension, but surface area only as the square. So as animals get bigger they have trouble radiating heat. That's why mice and rabbits are furry and elephants and hippos aren't. You can't make a mouse by scaling down an elephant.

YC represents a new, smaller kind of animal—so much smaller that all the rules are different.

Before us, most companies in the startup funding business were venture capital funds. VCs generally fund later stage companies than we do. And they supply so much money that, even though the other things they do may be very valuable, it's not that inaccurate to regard VCs as sources of money. Good VCs are "smart money," but they're still money.

All good investors supply a combination of money and help. But these scale differently, just as volume and surface area do. Late stage investors supply huge amounts of money and comparatively little help: when a company about to go public gets a mezzanine round of $50 million, the deal tends to be almost entirely about money. As you move earlier in the venture funding process, the ratio of help to money increases, because earlier stage companies have different needs. Early stage companies need less money because they're smaller and cheaper to run, but they need more help because life is so precarious for them. So when VCs do a series A round for, say, $2 million, they generally expect to offer a significant amount of help along with the money.

Y Combinator occupies the earliest end of the spectrum. We're at least one and generally two steps before VC funding. (Though some startups go straight from YC to VC, the most common trajectory is to do an angel round first.) And what happens at Y Combinator is as different from what happens in a series A round as a series A round is from a mezzanine financing.

At our end, money is almost a negligible factor. The startup usually consists of just the founders. Their living expenses are the company's main expense, and since most founders are under 30, their living expenses are low. But at this early stage companies need a lot of help. Practically every question is still unanswered. Some companies we've funded have been working on their software for a year or more, but others haven't decided what to work on, or even who the founders should be.

When PR people and journalists recount the histories of startups after they've become big, they always underestimate how uncertain things were at first. They're not being deliberately misleading. When you look at a company like Google, it's hard to imagine they could once have been small and helpless. Sure, at one point they were a just a couple guys in a garage—but even then their greatness was assured, and all they had to do was roll forward along the railroad tracks of destiny.

Far from it. A lot of startups with just as promising beginnings end up failing. Google has such momentum now that it would be hard for anyone to stop them. But all it would have taken in the beginning would have been for two Google employees to focus on the wrong things for six months, and the company could have died.

We know, because we've been there, just how vulnerable startups are in the earliest phases. Curiously enough, that's why founders tend to get so rich from them. Reward is always proportionate to risk, and very early stage startups are insanely risky.

What we really do at Y Combinator is get startups launched straight. One of many metaphors you could use for YC is a steam catapult on an aircraft carrier. We get startups airborne. Barely airborne, but enough that they can accelerate fast.

When you're launching planes they have to be set up properly or you're just launching projectiles. They have to be pointed straight down the deck; the wings have to be trimmed properly; the engines have to be at full power; the pilot has to be ready. These are the kind of problems we deal with. After we fund startups we work closely with them for three months—so closely in fact that we insist they move to where we are. And what we do in those three months is make sure everything is set up for launch. If there are tensions between cofounders we help sort them out. We get all the paperwork set up properly so there are no nasty surprises later. If the founders aren't sure what to focus on first, we try to figure that out. If there is some obstacle right in front of them, we either try to remove it, or shift the startup sideways. The goal is to get every distraction out of the way so the founders can use that time to build (or finish building) something impressive. And then near the end of the three months we push the button on the steam catapult in the form of Demo Day, where the current group of startups present to pretty much every investor in Silicon Valley.

Launching companies isn't identical with launching products. Though we do spend a lot of time on launch strategies for products, there are some things that take too long to build for a startup to launch them before raising their next round of funding. Several of the most promising startups we've funded haven't launched their products yet, but are definitely launched as companies.

In the earliest stage, startups not only have more questions to answer, but they tend to be different kinds of questions. In later stage startups the questions are about deals, or hiring, or organization. In the earliest phase they tend to be about technology and design. What do you make? That's the first problem to solve. That's why our motto is "Make something people want." This is always a good thing for companies to do, but it's even more important early on, because it sets the bounds for every other question. Who you hire, how much money you raise, how you market yourself—they all depend on what you're making.

Because the early problems are so much about technology and design, you probably need to be hackers to do what we do. While some VCs have technical backgrounds, I don't know any who still write code. Their expertise is mostly in business—as it should be, because that's the kind of expertise you need in the phase between series A and (if you're lucky) IPO.

We're so different from VCs that we're really a different kind of animal. Can we claim founders are better off as a result of this new type of venture firm? I'm pretty sure the answer is yes, because YC is an improved version of what happened to our startup, and our case was not atypical. We started Viaweb with $10,000 in seed money from our friend Julian. He was a lawyer and arranged all our paperwork, so we could just code. We spent three months building a version 1, which we then presented to investors to raise more money. Sounds familiar, doesn't it? But YC improves on that significantly. Julian knew a lot about law and business, but his advice ended there; he was not a startup guy. So we made some basic mistakes early on. And when we presented to investors, we presented to only 2, because that was all we knew. If we'd had our later selves to encourage and

advise us, and Demo Day to present at, we would have been in much better shape. We probably could have raised money at 3 to 5 times the valuation we did.

If we take 7% of a company we fund, the founders only have to do 7.5% better in their next round of funding to end up net ahead. We certainly manage that.

So who is our 7% coming out of? If the founders end up net ahead it's not coming out of them. So is it coming out of later stage investors? Well, they do end up paying more. But I think they pay more because the company is actually more valuable. And later stage investors have no problem with that. The returns of a VC fund depend on the quality of the companies they invest in, not how cheaply they can buy stock in them.

If what we do is useful, why wasn't anyone doing it before? There are two answers to that. One is that people were doing it before, just haphazardly on a smaller scale. Before us, seed funding came primarily from individual angel investors. Larry and Sergey, for example, got their seed funding from Andy Bechtolsheim, one of the founders of Sun. And because he was a startup guy he probably gave them useful advice. But raising money from angel investors is a hit or miss thing. It's a sideline for most of them, so they only do a handful of deals a year and they don't spend a lot of time on the startups they invest in. And they're hard to reach, because they don't want random startups pestering them with business plans. The Google guys were lucky because they knew someone who knew Bechtolsheim. It generally takes a personal introduction with angels.

The other reason no one was doing quite what we do is that till recently it was a lot more expensive to start a startup. You'll notice we haven't funded any biotech startups. That's still expensive. But advancing technology has made web startups so cheap that you really can get a company airborne for $15,000. If you understand how to operate a steam catapult, at least.

So in effect what's happened is that a new ecological niche has opened up, and Y Combinator is the new kind of animal that has moved into it. We're not a replacement for venture capital funds. We occupy a new, adjacent niche. And conditions in our niche are really quite different. It's not just that the problems we face are different; the whole structure of the business is different. VCs are playing a zero-sum game. They're all competing for a slice of a fixed amount of "deal flow," and that explains a lot of their behavior. Whereas our m.o. is to create new deal flow, by encouraging hackers who would have gotten jobs to start their own startups instead. We compete more with employers than VCs.

It's not surprising something like this would happen. Most fields become more specialized—more articulated—as they develop, and startups are certainly an area in which there has been a lot of development over the past couple decades. The venture business in its present form is only about forty years old. It stands to reason it would evolve.

And it's natural that the new niche would at first be described, even by its

inhabitants, in terms of the old one. But really Y Combinator is not in the startup funding business. Really we're more of a small, furry steam catapult.

Thanks to Trevor Blackwell, Jessica Livingston, and Robert Morris for reading drafts of this.

 Comment on this essay.

You Weren't Meant to Have a Boss

March 2008, rev. June 2008

Technology tends to separate normal from natural. Our bodies weren't designed to eat the foods that people in rich countries eat, or to get so little exercise. There may be a similar problem with the way we work: a normal job may be as bad for us intellectually as white flour or sugar is for us physically.

I began to suspect this after spending several years working with startup founders. I've now worked with over 200 of them, and I've noticed a definite difference between programmers working on their own startups and those working for large organizations. I wouldn't say founders seem happier, necessarily; starting a startup can be very stressful. Maybe the best way to put it is to say that they're happier in the sense that your body is happier during a long run than sitting on a sofa eating doughnuts.

Though they're statistically abnormal, startup founders seem to be working in a way that's more natural for humans.

I was in Africa last year and saw a lot of animals in the wild that I'd only seen in zoos before. It was remarkable how different they seemed. Particularly lions. Lions in the wild seem about ten times more alive. They're like different animals. I suspect that working for oneself feels better to humans in much the same way that living in the wild must feel better to a wide-ranging predator like a lion. Life in a zoo is easier, but it isn't the life they were designed for.

Trees

What's so unnatural about working for a big company? The root of the problem is that humans weren't meant to work in such large groups.

Another thing you notice when you see animals in the wild is that each species

thrives in groups of a certain size. A herd of impalas might have 100 adults; baboons maybe 20; lions rarely 10. Humans also seem designed to work in groups, and what I've read about hunter-gatherers accords with research on organizations and my own experience to suggest roughly what the ideal size is: groups of 8 work well; by 20 they're getting hard to manage; and a group of 50 is really unwieldy. [1]

Whatever the upper limit is, we are clearly not meant to work in groups of several hundred. And yet—for reasons having more to do with technology than human nature—a great many people work for companies with hundreds or thousands of employees.

Companies know groups that large wouldn't work, so they divide themselves into units small enough to work together. But to coordinate these they have to introduce something new: bosses.

These smaller groups are always arranged in a tree structure. Your boss is the point where your group attaches to the tree. But when you use this trick for dividing a large group into smaller ones, something strange happens that I've never heard anyone mention explicitly. In the group one level up from yours, your boss represents your entire group. A group of 10 managers is not merely a group of 10 people working together in the usual way. It's really a group of groups. Which means for a group of 10 managers to work together as if they were simply a group of 10 individuals, the group working for each manager would have to work as if they were a single person—the workers and manager would each share only one person's worth of freedom between them.

In practice a group of people are never able to act as if they were one person. But in a large organization divided into groups in this way, the pressure is always in that direction. Each group tries its best to work as if it were the small group of individuals that humans were designed to work in. That was the point of creating it. And when you propagate that constraint, the result is that each person gets freedom of action in inverse proportion to the size of the entire tree. [2]

Anyone who's worked for a large organization has felt this. You can feel the difference between working for a company with 100 employees and one with 10,000, even if your group has only 10 people.

Corn Syrup

A group of 10 people within a large organization is a kind of fake tribe. The number of people you interact with is about right. But something is missing: individual initiative. Tribes of hunter-gatherers have much more freedom. The leaders have a little more power than other members of the tribe, but they don't generally tell them what to do and when the way a boss can.

It's not your boss's fault. The real problem is that in the group above you in the hierarchy, your entire group is one virtual person. Your boss is just the way that

constraint is imparted to you.

So working in a group of 10 people within a large organization feels both right and wrong at the same time. On the surface it feels like the kind of group you're meant to work in, but something major is missing. A job at a big company is like high fructose corn syrup: it has some of the qualities of things you're meant to like, but is disastrously lacking in others.

Indeed, food is an excellent metaphor to explain what's wrong with the usual sort of job.

For example, working for a big company is the default thing to do, at least for programmers. How bad could it be? Well, food shows that pretty clearly. If you were dropped at a random point in America today, nearly all the food around you would be bad for you. Humans were not designed to eat white flour, refined sugar, high fructose corn syrup, and hydrogenated vegetable oil. And yet if you analyzed the contents of the average grocery store you'd probably find these four ingredients accounted for most of the calories. "Normal" food is terribly bad for you. The only people who eat what humans were actually designed to eat are a few Birkenstock-wearing weirdos in Berkeley.

If "normal" food is so bad for us, why is it so common? There are two main reasons. One is that it has more immediate appeal. You may feel lousy an hour after eating that pizza, but eating the first couple bites feels great. The other is economies of scale. Producing junk food scales; producing fresh vegetables doesn't. Which means (a) junk food can be very cheap, and (b) it's worth spending a lot to market it.

If people have to choose between something that's cheap, heavily marketed, and appealing in the short term, and something that's expensive, obscure, and appealing in the long term, which do you think most will choose?

It's the same with work. The average MIT graduate wants to work at Google or Microsoft, because it's a recognized brand, it's safe, and they'll get paid a good salary right away. It's the job equivalent of the pizza they had for lunch. The drawbacks will only become apparent later, and then only in a vague sense of malaise.

And founders and early employees of startups, meanwhile, are like the Birkenstock-wearing weirdos of Berkeley: though a tiny minority of the population, they're the ones living as humans are meant to. In an artificial world, only extremists live naturally.

Programmers

The restrictiveness of big company jobs is particularly hard on programmers, because the essence of programming is to build new things. Sales people make much the same pitches every day; support people answer much the same

questions; but once you've written a piece of code you don't need to write it again. So a programmer working as programmers are meant to is always making new things. And when you're part of an organization whose structure gives each person freedom in inverse proportion to the size of the tree, you're going to face resistance when you do something new.

This seems an inevitable consequence of bigness. It's true even in the smartest companies. I was talking recently to a founder who considered starting a startup right out of college, but went to work for Google instead because he thought he'd learn more there. He didn't learn as much as he expected. Programmers learn by doing, and most of the things he wanted to do, he couldn't—sometimes because the company wouldn't let him, but often because the company's code wouldn't let him. Between the drag of legacy code, the overhead of doing development in such a large organization, and the restrictions imposed by interfaces owned by other groups, he could only try a fraction of the things he would have liked to. He said he has learned much more in his own startup, despite the fact that he has to do all the company's errands as well as programming, because at least when he's programming he can do whatever he wants.

An obstacle downstream propagates upstream. If you're not allowed to implement new ideas, you stop having them. And vice versa: when you can do whatever you want, you have more ideas about what to do. So working for yourself makes your brain more powerful in the same way a low-restriction exhaust system makes an engine more powerful.

Working for yourself doesn't have to mean starting a startup, of course. But a programmer deciding between a regular job at a big company and their own startup is probably going to learn more doing the startup.

You can adjust the amount of freedom you get by scaling the size of company you work for. If you start the company, you'll have the most freedom. If you become one of the first 10 employees you'll have almost as much freedom as the founders. Even a company with 100 people will feel different from one with 1000.

Working for a small company doesn't ensure freedom. The tree structure of large organizations sets an upper bound on freedom, not a lower bound. The head of a small company may still choose to be a tyrant. The point is that a large organization is compelled by its structure to be one.

Consequences

That has real consequences for both organizations and individuals. One is that companies will inevitably slow down as they grow larger, no matter how hard they try to keep their startup mojo. It's a consequence of the tree structure that every large organization is forced to adopt.

Or rather, a large organization could only avoid slowing down if they avoided tree structure. And since human nature limits the size of group that can work together,

the only way I can imagine for larger groups to avoid tree structure would be to have no structure: to have each group actually be independent, and to work together the way components of a market economy do.

That might be worth exploring. I suspect there are already some highly partitionable businesses that lean this way. But I don't know any technology companies that have done it.

There is one thing companies can do short of structuring themselves as sponges: they can stay small. If I'm right, then it really pays to keep a company as small as it can be at every stage. Particularly a technology company. Which means it's doubly important to hire the best people. Mediocre hires hurt you twice: they get less done, but they also make you big, because you need more of them to solve a given problem.

For individuals the upshot is the same: aim small. It will always suck to work for large organizations, and the larger the organization, the more it will suck.

In an essay I wrote a couple years ago I advised graduating seniors to work for a couple years for another company before starting their own. I'd modify that now. Work for another company if you want to, but only for a small one, and if you want to start your own startup, go ahead.

The reason I suggested college graduates not start startups immediately was that I felt most would fail. And they will. But ambitious programmers are better off doing their own thing and failing than going to work at a big company. Certainly they'll learn more. They might even be better off financially. A lot of people in their early twenties get into debt, because their expenses grow even faster than the salary that seemed so high when they left school. At least if you start a startup and fail your net worth will be zero rather than negative. [3]

We've now funded so many different types of founders that we have enough data to see patterns, and there seems to be no benefit from working for a big company. The people who've worked for a few years do seem better than the ones straight out of college, but only because they're that much older.

The people who come to us from big companies often seem kind of conservative. It's hard to say how much is because big companies made them that way, and how much is the natural conservatism that made them work for the big companies in the first place. But certainly a large part of it is learned. I know because I've seen it burn off.

Having seen that happen so many times is one of the things that convinces me that working for oneself, or at least for a small group, is the natural way for programmers to live. Founders arriving at Y Combinator often have the downtrodden air of refugees. Three months later they're transformed: they have so much more confidence that they seem as if they've grown several inches taller. [4] Strange as this sounds, they seem both more worried and happier at the same

time. Which is exactly how I'd describe the way lions seem in the wild.

Watching employees get transformed into founders makes it clear that the difference between the two is due mostly to environment—and in particular that the environment in big companies is toxic to programmers. In the first couple weeks of working on their own startup they seem to come to life, because finally they're working the way people are meant to.

Notes

[1] When I talk about humans being meant or designed to live a certain way, I mean by evolution.

[2] It's not only the leaves who suffer. The constraint propagates up as well as down. So managers are constrained too; instead of just doing things, they have to act through subordinates.

[3] Do not finance your startup with credit cards. Financing a startup with debt is usually a stupid move, and credit card debt stupidest of all. Credit card debt is a bad idea, period. It is a trap set by evil companies for the desperate and the foolish.

[4] The founders we fund used to be younger (initially we encouraged undergrads to apply), and the first couple times I saw this I used to wonder if they were actually getting physically taller.

Thanks to Trevor Blackwell, Ross Boucher, Aaron Iba, Abby Kirigin, Ivan Kirigin, Jessica Livingston, and Robert Morris for reading drafts of this.

- French Translation

- Russian Translation

How to Disagree

March 2008

The web is turning writing into a conversation. Twenty years ago, writers wrote and readers read. The web lets readers respond, and increasingly they do—in comment threads, on forums, and in their own blog posts.

Many who respond to something disagree with it. That's to be expected. Agreeing tends to motivate people less than disagreeing. And when you agree there's less to say. You could expand on something the author said, but he has probably already explored the most interesting implications. When you disagree you're entering territory he may not have explored.

The result is there's a lot more disagreeing going on, especially measured by the word. That doesn't mean people are getting angrier. The structural change in the way we communicate is enough to account for it. But though it's not anger that's driving the increase in disagreement, there's a danger that the increase in disagreement will make people angrier. Particularly online, where it's easy to say things you'd never say face to face.

If we're all going to be disagreeing more, we should be careful to do it well. What does it mean to disagree well? Most readers can tell the difference between mere name-calling and a carefully reasoned refutation, but I think it would help to put names on the intermediate stages. So here's an attempt at a disagreement hierarchy:

DH0. Name-calling.

This is the lowest form of disagreement, and probably also the most common. We've all seen comments like this:

> u r a fag!!!!!!!!!!

But it's important to realize that more articulate name-calling has just as little weight. A comment like

> The author is a self-important dilettante.

is really nothing more than a pretentious version of "u r a fag."

DH1. Ad Hominem.

An ad hominem attack is not quite as weak as mere name-calling. It might actually carry some weight. For example, if a senator wrote an article saying senators' salaries should be increased, one could respond:

> Of course he would say that. He's a senator.

This wouldn't refute the author's argument, but it may at least be relevant to the case. It's still a very weak form of disagreement, though. If there's something wrong with the senator's argument, you should say what it is; and if there isn't, what difference does it make that he's a senator?

Saying that an author lacks the authority to write about a topic is a variant of ad hominem—and a particularly useless sort, because good ideas often come from outsiders. The question is whether the author is correct or not. If his lack of authority caused him to make mistakes, point those out. And if it didn't, it's not a problem.

DH2. Responding to Tone.

The next level up we start to see responses to the writing, rather than the writer. The lowest form of these is to disagree with the author's tone. E.g.

> I can't believe the author dismisses intelligent design in such a cavalier fashion.

Though better than attacking the author, this is still a weak form of disagreement. It matters much more whether the author is wrong or right than what his tone is. Especially since tone is so hard to judge. Someone who has a chip on their shoulder about some topic might be offended by a tone that to other readers seemed neutral.

So if the worst thing you can say about something is to criticize its tone, you're not saying much. Is the author flippant, but correct? Better that than grave and wrong. And if the author is incorrect somewhere, say where.

DH3. Contradiction.

In this stage we finally get responses to what was said, rather than how or by whom. The lowest form of response to an argument is simply to state the opposing case, with little or no supporting evidence.

This is often combined with DH2 statements, as in:

> I can't believe the author dismisses intelligent design in such a cavalier fashion. Intelligent design is a legitimate scientific theory.

Contradiction can sometimes have some weight. Sometimes merely seeing the opposing case stated explicitly is enough to see that it's right. But usually evidence

will help.

DH4. Counterargument.

At level 4 we reach the first form of convincing disagreement: counterargument. Forms up to this point can usually be ignored as proving nothing. Counterargument might prove something. The problem is, it's hard to say exactly what.

Counterargument is contradiction plus reasoning and/or evidence. When aimed squarely at the original argument, it can be convincing. But unfortunately it's common for counterarguments to be aimed at something slightly different. More often than not, two people arguing passionately about something are actually arguing about two different things. Sometimes they even agree with one another, but are so caught up in their squabble they don't realize it.

There could be a legitimate reason for arguing against something slightly different from what the original author said: when you feel they missed the heart of the matter. But when you do that, you should say explicitly you're doing it.

DH5. Refutation.

The most convincing form of disagreement is refutation. It's also the rarest, because it's the most work. Indeed, the disagreement hierarchy forms a kind of pyramid, in the sense that the higher you go the fewer instances you find.

To refute someone you probably have to quote them. You have to find a "smoking gun," a passage in whatever you disagree with that you feel is mistaken, and then explain why it's mistaken. If you can't find an actual quote to disagree with, you may be arguing with a straw man.

While refutation generally entails quoting, quoting doesn't necessarily imply refutation. Some writers quote parts of things they disagree with to give the appearance of legitimate refutation, then follow with a response as low as DH3 or even DH0.

DH6. Refuting the Central Point.

The force of a refutation depends on what you refute. The most powerful form of disagreement is to refute someone's central point.

Even as high as DH5 we still sometimes see deliberate dishonesty, as when someone picks out minor points of an argument and refutes those. Sometimes the spirit in which this is done makes it more of a sophisticated form of ad hominem than actual refutation. For example, correcting someone's grammar, or harping on minor mistakes in names or numbers. Unless the opposing argument actually depends on such things, the only purpose of correcting them is to discredit one's opponent.

Truly refuting something requires one to refute its central point, or at least one of them. And that means one has to commit explicitly to what the central point is. So a truly effective refutation would look like:

> The author's main point seems to be x. As he says:
>
> > <quotation>
>
> But this is wrong for the following reasons...

The quotation you point out as mistaken need not be the actual statement of the author's main point. It's enough to refute something it depends upon.

What It Means

Now we have a way of classifying forms of disagreement. What good is it? One thing the disagreement hierarchy *doesn't* give us is a way of picking a winner. DH levels merely describe the form of a statement, not whether it's correct. A DH6 response could still be completely mistaken.

But while DH levels don't set a lower bound on the convincingness of a reply, they do set an upper bound. A DH6 response might be unconvincing, but a DH2 or lower response is always unconvincing.

The most obvious advantage of classifying the forms of disagreement is that it will help people to evaluate what they read. In particular, it will help them to see through intellectually dishonest arguments. An eloquent speaker or writer can give the impression of vanquishing an opponent merely by using forceful words. In fact that is probably the defining quality of a demagogue. By giving names to the different forms of disagreement, we give critical readers a pin for popping such balloons.

Such labels may help writers too. Most intellectual dishonesty is unintentional. Someone arguing against the tone of something he disagrees with may believe he's really saying something. Zooming out and seeing his current position on the disagreement hierarchy may inspire him to try moving up to counterargument or refutation.

But the greatest benefit of disagreeing well is not just that it will make conversations better, but that it will make the people who have them happier. If you study conversations, you find there is a lot more meanness down in DH1 than up in DH6. You don't have to be mean when you have a real point to make. In fact, you don't want to. If you have something real to say, being mean just gets in the way.

If moving up the disagreement hierarchy makes people less mean, that will make most of them happier. Most people don't really enjoy being mean; they do it because they can't help it.

Thanks to Trevor Blackwell and Jessica Livingston for reading drafts of this.

Related:

- What You Can't Say
- The Age of the Essay

- Italian Translation
- Russian Translation

- Swedish Translation
- Spanish Translation

- German Translation
- French Translation

- Arabic Translation
- Finnish Translation

- Italian Translation
- Turkish Translation

Some Heroes

April 2008

There are some topics I save up because they'll be so much fun to write about. This is one of them: a list of my heroes.

I'm not claiming this is a list of the *n* most admirable people. Who could make such a list, even if they wanted to?

Einstein isn't on the list, for example, even though he probably deserves to be on any shortlist of admirable people. I once asked a physicist friend if Einstein was really as smart as his fame implies, and she said that yes, he was. So why isn't he on the list? Because I had to ask. This is a list of people who've influenced me, not people who would have if I understood their work.

My test was to think of someone and ask "is this person my hero?" It often returned surprising answers. For example, it returned false for Montaigne, who was arguably the inventor of the essay. Why? When I thought about what it meant to call someone a hero, it meant I'd decide what to do by asking what they'd do in the same situation. That's a stricter standard than admiration.

After I made the list, I looked to see if there was a pattern, and there was, a very clear one. Everyone on the list had two qualities: they cared almost excessively about their work, and they were absolutely honest. By honest I don't mean trustworthy so much as that they never pander: they never say or do something because that's what the audience wants. They are all fundamentally subversive for this reason, though they conceal it to varying degrees.

Jack Lambert

I grew up in Pittsburgh in the 1970s. Unless you were there it's hard to imagine how that town felt about the Steelers. Locally, all the news was bad. The steel industry was dying. But the Steelers were the best team in football — and moreover, in a way that seemed to reflect the personality of the city. They didn't do anything fancy. They just got the job done.

Other players were more famous: Terry Bradshaw, Franco Harris, Lynn Swann. But they played offense, and you always get more attention for that. It seemed to me as a twelve year old football expert that the best of them all was Jack Lambert.

And what made him so good was that he was utterly relentless. He didn't just care about playing well; he cared almost too much. He seemed to regard it as a personal insult when someone from the other team had possession of the ball on his side of the line of scrimmage.

The suburbs of Pittsburgh in the 1970s were a pretty dull place. School was boring. All the adults around were bored with their jobs working for big companies. Everything that came to us through the mass media was (a) blandly uniform and (b) produced elsewhere. Jack Lambert was the exception. He was like nothing else I'd seen.

Kenneth Clark

Kenneth Clark is the best nonfiction writer I know of, on any subject. Most people who write about art history don't really like art; you can tell from a thousand little signs. But Clark did, and not just intellectually, but the way one anticipates a delicious dinner.

What really makes him stand out, though, is the quality of his ideas. His style is deceptively casual, but there is more in his books than in a library of art monographs. Reading *The Nude* is like a ride in a Ferrari. Just as you're getting settled, you're slammed back in your seat by the acceleration. Before you can adjust, you're thrown sideways as the car screeches into the first turn. His brain throws off ideas almost too fast to grasp them. Finally at the end of the chapter you come to a halt, with your eyes wide and a big smile on your face.

Kenneth Clark was a star in his day, thanks to the documentary series *Civilisation*. And if you read only one book about art history, *Civilisation* is the one I'd recommend. It's much better than the drab Sears Catalogs of art that undergraduates are forced to buy for Art History 101.

Larry Mihalko

A lot of people have a great teacher at some point in their childhood. Larry Mihalko was mine. When I look back it's like there's a line drawn between third and fourth grade. After Mr. Mihalko, everything was different.

Why? First of all, he was intellectually curious. I had a few other teachers who were smart, but I wouldn't describe them as intellectually curious. In retrospect, he was out of place as an elementary school teacher, and I think he knew it. That must have been hard for him, but it was wonderful for us, his students. His class was a constant adventure. I used to like going to school every day.

The other thing that made him different was that he liked us. Kids are good at telling that. The other teachers were at best benevolently indifferent. But Mr. Mihalko seemed like he actually wanted to be our friend. On the last day of fourth grade, he got out one of the heavy school record players and played James Taylor's "You've Got a Friend" to us. Just call out my name, and you know wherever I am,

I'll come running. He died at 59 of lung cancer. I've never cried like I cried at his funeral.

Leonardo

One of the things I've learned about making things that I didn't realize when I was a kid is that much of the best stuff isn't made for audiences, but for oneself. You see paintings and drawings in museums and imagine they were made for you to look at. Actually a lot of the best ones were made as a way of exploring the world, not as a way to please other people. The best of these explorations are sometimes more pleasing than stuff made explicitly to please.

Leonardo did a lot of things. One of his most admirable qualities was that he did so many different things that were admirable. What people know of him now is his paintings and his more flamboyant inventions, like flying machines. That makes him seem like some kind of dreamer who sketched artists' conceptions of rocket ships on the side. In fact he made a large number of far more practical technical discoveries. He was as good an engineer as a painter.

His most impressive work, to me, is his drawings. They're clearly made more as a way of studying the world than producing something beautiful. And yet they can hold their own with any work of art ever made. No one else, before or since, was that good when no one was looking.

Robert Morris

Robert Morris has a very unusual quality: he's never wrong. It might seem this would require you to be omniscient, but actually it's surprisingly easy. Don't say anything unless you're fairly sure of it. If you're not omniscient, you just don't end up saying much.

More precisely, the trick is to pay careful attention to how you qualify what you say. By using this trick, Robert has, as far as I know, managed to be mistaken only once, and that was when he was an undergrad. When the Mac came out, he said that little desktop computers would never be suitable for real hacking.

It's wrong to call it a trick in his case, though. If it were a conscious trick, he would have slipped in a moment of excitement. With Robert this quality is wired-in. He has an almost superhuman integrity. He's not just generally correct, but also correct about how correct he is.

You'd think it would be such a great thing never to be wrong that everyone would do this. It doesn't seem like that much extra work to pay as much attention to the error on an idea as to the idea itself. And yet practically no one does. I know how hard it is, because since meeting Robert I've tried to do in software what he seems to do in hardware.

P. G. Wodehouse

People are finally starting to admit that Wodehouse was a great writer. If you want to be thought a great novelist in your own time, you have to sound intellectual. If what you write is popular, or entertaining, or funny, you're ipso facto suspect. That makes Wodehouse doubly impressive, because it meant that to write as he wanted to, he had to commit to being despised in his own lifetime.

Evelyn Waugh called him a great writer, but to most people at the time that would have read as a chivalrous or deliberately perverse gesture. At the time any random autobiographical novel by a recent college grad could count on more respectful treatment from the literary establishment.

Wodehouse may have begun with simple atoms, but the way he composed them into molecules was near faultless. His rhythm in particular. It makes me self-conscious to write about it. I can think of only two other writers who came near him for style: Evelyn Waugh and Nancy Mitford. Those three used the English language like they owned it.

But Wodehouse has something neither of them did. He's at ease. Evelyn Waugh and Nancy Mitford cared what other people thought of them: he wanted to seem aristocratic; she was afraid she wasn't smart enough. But Wodehouse didn't give a damn what anyone thought of him. He wrote exactly what he wanted.

Alexander Calder

Calder's on this list because he makes me happy. Can his work stand up to Leonardo's? Probably not. There might not be anything from the 20th Century that can. But what was good about Modernism, Calder had, and had in a way that he made seem effortless.

What was good about Modernism was its freshness. Art became stuffy in the nineteenth century. The paintings that were popular at the time were mostly the art equivalent of McMansions—big, pretentious, and fake. Modernism meant starting over, making things with the same earnest motives that children might. The artists who benefited most from this were the ones who had preserved a child's confidence, like Klee and Calder.

Klee was impressive because he could work in so many different styles. But between the two I like Calder better, because his work seemed happier. Ultimately the point of art is to engage the viewer. It's hard to predict what will; often something that seems interesting at first will bore you after a month. Calder's sculptures never get boring. They just sit there quietly radiating optimism, like a battery that never runs out. As far as I can tell from books and photographs, the happiness of Calder's work is his own happiness showing through.

Jane Austen

Everyone admires Jane Austen. Add my name to the list. To me she seems the

best novelist of all time.

I'm interested in how things work. When I read most novels, I pay as much attention to the author's choices as to the story. But in her novels I can't see the gears at work. Though I'd really like to know how she does what she does, I can't figure it out, because she's so good that her stories don't seem made up. I feel like I'm reading a description of something that actually happened.

I used to read a lot of novels when I was younger. I can't read most anymore, because they don't have enough information in them. Novels seem so impoverished compared to history and biography. But reading Austen is like reading nonfiction. She writes so well you don't even notice her.

John McCarthy

John McCarthy invented Lisp, the field of (or at least the term) artificial intelligence, and was an early member of both of the top two computer science departments, MIT and Stanford. No one would dispute that he's one of the greats, but he's an especial hero to me because of Lisp.

It's hard for us now to understand what a conceptual leap that was at the time. Paradoxically, one of the reasons his achievement is hard to appreciate is that it was so successful. Practically every programming language invented in the last 20 years includes ideas from Lisp, and each year the median language gets more Lisplike.

In 1958 these ideas were anything but obvious. In 1958 there seem to have been two ways of thinking about programming. Some people thought of it as math, and proved things about Turing Machines. Others thought of it as a way to get things done, and designed languages all too influenced by the technology of the day. McCarthy alone bridged the gap. He designed a language that was math. But designed is not really the word; discovered is more like it.

The Spitfire

As I was making this list I found myself thinking of people like Douglas Bader and R.J. Mitchell and Jeffrey Quill and I realized that though all of them had done many things in their lives, there was one factor above all that connected them: the Spitfire.

This is supposed to be a list of heroes. How can a machine be on it? Because that machine was not just a machine. It was a lens of heroes. Extraordinary devotion went into it, and extraordinary courage came out.

It's a cliche to call World War II a contest between good and evil, but between fighter designs, it really was. The Spitfire's original nemesis, the ME 109, was a brutally practical plane. It was a killing machine. The Spitfire was optimism embodied. And not just in its beautiful lines: it was at the edge of what could be

manufactured. But taking the high road worked. In the air, beauty had the edge, just.

Steve Jobs

People alive when Kennedy was killed usually remember exactly where they were when they heard about it. I remember exactly where I was when a friend asked if I'd heard Steve Jobs had cancer. It was like the floor dropped out. A few seconds later she told me that it was a rare operable type, and that he'd be ok. But those seconds seemed long.

I wasn't sure whether to include Jobs on this list. A lot of people at Apple seem to be afraid of him, which is a bad sign. But he compels admiration.

There's no name for what Steve Jobs is, because there hasn't been anyone quite like him before. He doesn't design Apple's products himself. Historically the closest analogy to what he does are the great Renaissance patrons of the arts. As the CEO of a company, that makes him unique.

Most CEOs delegate taste to a subordinate. The design paradox means they're choosing more or less at random. But Steve Jobs actually has taste himself — such good taste that he's shown the world how much more important taste is than they realized.

Isaac Newton

Newton has a strange role in my pantheon of heroes: he's the one I reproach myself with. He worked on big things, at least for part of his life. It's so easy to get distracted working on small stuff. The questions you're answering are pleasantly familiar. You get immediate rewards — in fact, you get bigger rewards in your time if you work on matters of passing importance. But I'm uncomfortably aware that this is the route to well-deserved obscurity.

To do really great things, you have to seek out questions people didn't even realize were questions. There have probably been other people who did this as well as Newton, for their time, but Newton is my model of this kind of thought. I can just begin to understand what it must have felt like for him.

You only get one life. Why not do something huge? The phrase "paradigm shift" is overused now, but Kuhn was onto something. And you know more are out there, separated from us by what will later seem a surprisingly thin wall of laziness and stupidity. If we work like Newton.

Thanks to Trevor Blackwell, Jessica Livingston, and Jackie McDonough for reading drafts of this.

- Japanese Translation

Why There Aren't More Googles

April 2008

Umair Haque wrote recently that the reason there aren't more Googles is that most startups get bought before they can change the world.

> Google, despite serious interest from Microsoft and Yahoo—what must have seemed like lucrative interest at the time—didn't sell out. Google might simply have been nothing but Yahoo's or MSN's search box.
>
> Why isn't it? Because Google had a deeply felt sense of purpose: a conviction to change the world for the better.

This has a nice sound to it, but it isn't true. Google's founders were willing to sell early on. They just wanted more than acquirers were willing to pay.

It was the same with Facebook. They would have sold, but Yahoo blew it by offering too little.

Tip for acquirers: when a startup turns you down, consider raising your offer, because there's a good chance the outrageous price they want will later seem a bargain. [1]

From the evidence I've seen so far, startups that turn down acquisition offers usually end up doing better. Not always, but usually there's a bigger offer coming, or perhaps even an IPO.

Of course, the reason startups do better when they turn down acquisition offers is not necessarily that all such offers undervalue startups. More likely the reason is that the kind of founders who have the balls to turn down a big offer also tend to be very successful. That spirit is exactly what you want in a startup.

While I'm sure Larry and Sergey do want to change the world, at least now, the reason Google survived to become a big, independent company is the same reason Facebook has so far remained independent: acquirers underestimated them.

Corporate M&A is a strange business in that respect. They consistently lose the best deals, because turning down reasonable offers is the most reliable test you could invent for whether a startup will make it big.

VCs

So what's the real reason there aren't more Googles? Curiously enough, it's the same reason Google and Facebook have remained independent: money guys undervalue the most innovative startups.

The reason there aren't more Googles is not that investors encourage innovative startups to sell out, but that they won't even fund them. I've learned a lot about VCs during the 3 years we've been doing Y Combinator, because we often have to work quite closely with them. The most surprising thing I've learned is how conservative they are. VC firms present an image of boldly encouraging innovation. Only a handful actually do, and even they are more conservative in reality than you'd guess from reading their sites.

I used to think of VCs as piratical: bold but unscrupulous. On closer acquaintance they turn out to be more like bureaucrats. They're more upstanding than I used to think (the good ones, at least), but less bold. Maybe the VC industry has changed. Maybe they used to be bolder. But I suspect it's the startup world that has changed, not them. The low cost of starting a startup means the average good bet is a riskier one, but most existing VC firms still operate as if they were investing in hardware startups in 1985.

Howard Aiken said "Don't worry about people stealing your ideas. If your ideas are any good, you'll have to ram them down people's throats." I have a similar feeling when I'm trying to convince VCs to invest in startups Y Combinator has funded. They're terrified of really novel ideas, unless the founders are good enough salesmen to compensate.

But it's the bold ideas that generate the biggest returns. Any really good new idea will seem bad to most people; otherwise someone would already be doing it. And yet most VCs are driven by consensus, not just within their firms, but within the VC community. The biggest factor determining how a VC will feel about your startup is how other VCs feel about it. I doubt they realize it, but this algorithm guarantees they'll miss all the very best ideas. The more people who have to like a new idea, the more outliers you lose.

Whoever the next Google is, they're probably being told right now by VCs to come back when they have more "traction."

Why are VCs so conservative? It's probably a combination of factors. The large size of their investments makes them conservative. Plus they're investing other people's money, which makes them worry they'll get in trouble if they do something risky and it fails. Plus most of them are money guys rather than technical guys, so they don't understand what the startups they're investing in do.

What's Next

The exciting thing about market economies is that stupidity equals opportunity.

And so it is in this case. There is a huge, unexploited opportunity in startup investing. Y Combinator funds startups at the very beginning. VCs will fund them once they're already starting to succeed. But between the two there is a substantial gap.

There are companies that will give $20k to a startup that has nothing more than the founders, and there are companies that will give $2 million to a startup that's already taking off, but there aren't enough investors who will give $200k to a startup that seems very promising but still has some things to figure out. This territory is occupied mostly by individual angel investors—people like Andy Bechtolsheim, who gave Google $100k when they seemed promising but still had some things to figure out. I like angels, but there just aren't enough of them, and investing is for most of them a part time job.

And yet as it gets cheaper to start startups, this sparsely occupied territory is becoming more and more valuable. Nowadays a lot of startups don't want to raise multi-million dollar series A rounds. They don't need that much money, and they don't want the hassles that come with it. The median startup coming out of Y Combinator wants to raise $250-500k. When they go to VC firms they have to ask for more because they know VCs aren't interested in such small deals.

VCs are money managers. They're looking for ways to put large sums to work. But the startup world is evolving away from their current model.

Startups have gotten cheaper. That means they want less money, but also that there are more of them. So you can still get large returns on large amounts of money; you just have to spread it more broadly.

I've tried to explain this to VC firms. Instead of making one $2 million investment, make five $400k investments. Would that mean sitting on too many boards? Don't sit on their boards. Would that mean too much due diligence? Do less. If you're investing at a tenth the valuation, you only have to be a tenth as sure.

It seems obvious. But I've proposed to several VC firms that they set aside some money and designate one partner to make more, smaller bets, and they react as if I'd proposed the partners all get nose rings. It's remarkable how wedded they are to their standard m.o.

But there is a big opportunity here, and one way or the other it's going to get filled. Either VCs will evolve down into this gap or, more likely, new investors will appear to fill it. That will be a good thing when it happens, because these new investors will be compelled by the structure of the investments they make to be ten times bolder than present day VCs. And that will get us a lot more Googles. At least, as long as acquirers remain stupid.

Notes

[1] Another tip: If you want to get all that value, don't destroy the startup after you buy it. Give the founders enough autonomy that they can grow the acquisition into what it would have become.

Thanks to Sam Altman, Paul Buchheit, David Hornik, Jessica Livingston, Robert Morris, and Fred Wilson for reading drafts of this.

- Russian Translation

Be Good

April 2008

(This essay is derived from a talk at the 2008 Startup School.)

About a month after we started Y Combinator we came up with the phrase that became our motto: Make something people want. We've learned a lot since then, but if I were choosing now that's still the one I'd pick.

Another thing we tell founders is not to worry too much about the business model, at least at first. Not because making money is unimportant, but because it's so much easier than building something great.

A couple weeks ago I realized that if you put those two ideas together, you get something surprising. Make something people want. Don't worry too much about making money. What you've got is a description of a charity.

When you get an unexpected result like this, it could either be a bug or a new discovery. Either businesses aren't supposed to be like charities, and we've proven by reductio ad absurdum that one or both of the principles we began with is false. Or we have a new idea.

I suspect it's the latter, because as soon as this thought occurred to me, a whole bunch of other things fell into place.

Examples

For example, Craigslist. It's not a charity, but they run it like one. And they're astoundingly successful. When you scan down the list of most popular web sites, the number of employees at Craigslist looks like a misprint. Their revenues aren't

as high as they could be, but most startups would be happy to trade places with them.

In Patrick O'Brian's novels, his captains always try to get upwind of their opponents. If you're upwind, you decide when and if to engage the other ship. Craigslist is effectively upwind of enormous revenues. They'd face some challenges if they wanted to make more, but not the sort you face when you're tacking upwind, trying to force a crappy product on ambivalent users by spending ten times as much on sales as on development. [1]

I'm not saying startups should aim to end up like Craigslist. They're a product of unusual circumstances. But they're a good model for the early phases.

Google looked a lot like a charity in the beginning. They didn't have ads for over a year. At year 1, Google was indistinguishable from a nonprofit. If a nonprofit or government organization had started a project to index the web, Google at year 1 is the limit of what they'd have produced.

Back when I was working on spam filters I thought it would be a good idea to have a web-based email service with good spam filtering. I wasn't thinking of it as a company. I just wanted to keep people from getting spammed. But as I thought more about this project, I realized it would probably have to be a company. It would cost something to run, and it would be a pain to fund with grants and donations.

That was a surprising realization. Companies often claim to be benevolent, but it was surprising to realize there were purely benevolent projects that had to be embodied as companies to work.

I didn't want to start another company, so I didn't do it. But if someone had, they'd probably be quite rich now. There was a window of about two years when spam was increasing rapidly but all the big email services had terrible filters. If someone had launched a new, spam-free mail service, users would have flocked to it.

Notice the pattern here? From either direction we get to the same spot. If you start from successful startups, you find they often behaved like nonprofits. And if you start from ideas for nonprofits, you find they'd often make good startups.

Power

How wide is this territory? Would all good nonprofits be good companies? Possibly not. What makes Google so valuable is that their users have money. If you make people with money love you, you can probably get some of it. But could you also base a successful startup on behaving like a nonprofit to people who don't have money? Could you, for example, grow a successful startup out of curing an unfashionable but deadly disease like malaria?

I'm not sure, but I suspect that if you pushed this idea, you'd be surprised how far it would go. For example, people who apply to Y Combinator don't generally have much money, and yet we can profit by helping them, because with our help they could make money. Maybe the situation is similar with malaria. Maybe an organization that helped lift its weight off a country could benefit from the resulting growth.

I'm not proposing this is a serious idea. I don't know anything about malaria. But I've been kicking ideas around long enough to know when I come across a powerful one.

One way to guess how far an idea extends is to ask yourself at what point you'd bet against it. The thought of betting against benevolence is alarming in the same way as saying that something is technically impossible. You're just asking to be made a fool of, because these are such powerful forces. [2]

For example, initially I thought maybe this principle only applied to Internet startups. Obviously it worked for Google, but what about Microsoft? Surely Microsoft isn't benevolent? But when I think back to the beginning, they were. Compared to IBM they were like Robin Hood. When IBM introduced the PC, they thought they were going to make money selling hardware at high prices. But by gaining control of the PC standard, Microsoft opened up the market to any manufacturer. Hardware prices plummeted, and lots of people got to have computers who couldn't otherwise have afforded them. It's the sort of thing you'd expect Google to do.

Microsoft isn't so benevolent now. Now when one thinks of what Microsoft does to users, all the verbs that come to mind begin with F. [3] And yet it doesn't seem to pay. Their stock price has been flat for years. Back when they were Robin Hood, their stock price rose like Google's. Could there be a connection?

You can see how there would be. When you're small, you can't bully customers, so you have to charm them. Whereas when you're big you can maltreat them at will, and you tend to, because it's easier than satisfying them. You grow big by being nice, but you can stay big by being mean.

You get away with it till the underlying conditions change, and then all your victims escape. So "Don't be evil" may be the most valuable thing Paul Buchheit made for Google, because it may turn out to be an elixir of corporate youth. I'm sure they find it constraining, but think how valuable it will be if it saves them from lapsing into the fatal laziness that afflicted Microsoft and IBM.

The curious thing is, this elixir is freely available to any other company. Anyone can adopt "Don't be evil." The catch is that people will hold you to it. So I don't think you're going to see record labels or tobacco companies using this discovery.

Morale

There's a lot of external evidence that benevolence works. But how does it work? One advantage of investing in a large number of startups is that you get a lot of data about how they work. From what we've seen, being good seems to help startups in three ways: it improves their morale, it makes other people want to help them, and above all, it helps them be decisive.

Morale is tremendously important to a startup—so important that morale alone is almost enough to determine success. Startups are often described as emotional roller-coasters. One minute you're going to take over the world, and the next you're doomed. The problem with feeling you're doomed is not just that it makes you unhappy, but that it makes you *stop working*. So the downhills of the roller-coaster are more of a self fulfilling prophecy than the uphills. If feeling you're going to succeed makes you work harder, that probably improves your chances of succeeding, but if feeling you're going to fail makes you stop working, that practically guarantees you'll fail.

Here's where benevolence comes in. If you feel you're really helping people, you'll keep working even when it seems like your startup is doomed. Most of us have some amount of natural benevolence. The mere fact that someone needs you makes you want to help them. So if you start the kind of startup where users come back each day, you've basically built yourself a giant tamagotchi. You've made something you need to take care of.

Blogger is a famous example of a startup that went through really low lows and survived. At one point they ran out of money and everyone left. Evan Williams came in to work the next day, and there was no one but him. What kept him going? Partly that users needed him. He was hosting thousands of people's blogs. He couldn't just let the site die.

There are many advantages of launching quickly, but the most important may be that once you have users, the tamagotchi effect kicks in. Once you have users to take care of, you're forced to figure out what will make them happy, and that's actually very valuable information.

The added confidence that comes from trying to help people can also help you with investors. One of the founders of Chatterous told me recently that he and his cofounder had decided that this service was something the world needed, so they were going to keep working on it no matter what, even if they had to move back to Canada and live in their parents' basements.

Once they realized this, they stopped caring so much what investors thought about them. They still met with them, but they weren't going to die if they didn't get their money. And you know what? The investors got a lot more interested. They could sense that the Chatterouses were going to do this startup with or without them.

If you're really committed and your startup is cheap to run, you become very hard to kill. And practically all startups, even the most successful, come close to death

at some point. So if doing good for people gives you a sense of mission that makes you harder to kill, that alone more than compensates for whatever you lose by not choosing a more selfish project.

Help

Another advantage of being good is that it makes other people want to help you. This too seems to be an inborn trait in humans.

One of the startups we've funded, Octopart, is currently locked in a classic battle of good versus evil. They're a search site for industrial components. A lot of people need to search for components, and before Octopart there was no good way to do it. That, it turned out, was no coincidence.

Octopart built the right way to search for components. Users like it and they've been growing rapidly. And yet for most of Octopart's life, the biggest distributor, Digi-Key, has been trying to force them take their prices off the site. Octopart is sending them customers for free, and yet Digi-Key is trying to make that traffic stop. Why? Because their current business model depends on overcharging people who have incomplete information about prices. They don't want search to work.

The Octoparts are the nicest guys in the world. They dropped out of the PhD program in physics at Berkeley to do this. They just wanted to fix a problem they encountered in their research. Imagine how much time you could save the world's engineers if they could do searches online. So when I hear that a big, evil company is trying to stop them in order to keep search broken, it makes me really want to help them. It makes me spend more time on the Octoparts than I do with most of the other startups we've funded. It just made me spend several minutes telling you how great they are. Why? Because they're good guys and they're trying to help the world.

If you're benevolent, people will rally around you: investors, customers, other companies, and potential employees. In the long term the most important may be the potential employees. I think everyone knows now that good hackers are much better than mediocre ones. If you can attract the best hackers to work for you, as Google has, you have a big advantage. And the very best hackers tend to be idealistic. They're not desperate for a job. They can work wherever they want. So most want to work on things that will make the world better.

Compass

But the most important advantage of being good is that it acts as a compass. One of the hardest parts of doing a startup is that you have so many choices. There are just two or three of you, and a thousand things you could do. How do you decide?

Here's the answer: Do whatever's best for your users. You can hold onto this like a rope in a hurricane, and it will save you if anything can. Follow it and it will take you through everything you need to do.

It's even the answer to questions that seem unrelated, like how to convince investors to give you money. If you're a good salesman, you could try to just talk them into it. But the more reliable route is to convince them through your users: if you make something users love enough to tell their friends, you grow exponentially, and that will convince any investor.

Being good is a particularly useful strategy for making decisions in complex situations because it's stateless. It's like telling the truth. The trouble with lying is that you have to remember everything you've said in the past to make sure you don't contradict yourself. If you tell the truth you don't have to remember anything, and that's a really useful property in domains where things happen fast.

For example, Y Combinator has now invested in 80 startups, 57 of which are still alive. (The rest have died or merged or been acquired.) When you're trying to advise 57 startups, it turns out you have to have a stateless algorithm. You can't have ulterior motives when you have 57 things going on at once, because you can't remember them. So our rule is just to do whatever's best for the founders. Not because we're particularly benevolent, but because it's the only algorithm that works on that scale.

When you write something telling people to be good, you seem to be claiming to be good yourself. So I want to say explicitly that I am not a particularly good person. When I was a kid I was firmly in the camp of bad. The way adults used the word good, it seemed to be synonymous with quiet, so I grew up very suspicious of it.

You know how there are some people whose names come up in conversation and everyone says "He's *such* a great guy?" People never say that about me. The best I get is "he means well." I am not claiming to be good. At best I speak good as a second language.

So I'm not suggesting you be good in the usual sanctimonious way. I'm suggesting it because it works. It will work not just as a statement of "values," but as a guide to strategy, and even a design spec for software. Don't just not be evil. Be good.

Notes

[1] Fifty years ago it would have seemed shocking for a public company not to pay dividends. Now many tech companies don't. The markets seem to have figured out how to value potential dividends. Maybe that isn't the last step in this evolution. Maybe markets will eventually get comfortable with potential earnings. (VCs already are, and at least some of them consistently make money.)

I realize this sounds like the stuff one used to hear about the "new economy" during the Bubble. Believe me, I was not drinking that kool-aid at the time. But I'm convinced there were some good ideas buried in Bubble thinking. For example, it's ok to focus on growth instead of profits—but only if the growth is genuine. You can't be buying users; that's a pyramid scheme. But a company with rapid, genuine growth is valuable, and eventually markets learn how to value valuable things.

[2] The idea of starting a company with benevolent aims is currently undervalued, because the kind of people who currently make that their explicit goal don't usually do a very good job.

It's one of the standard career paths of trustafarians to start some vaguely benevolent business. The problem with most of them is that they either have a bogus political agenda or are feebly executed. The trustafarians' ancestors didn't get rich by preserving their traditional culture; maybe people in Bolivia don't want to either. And starting an organic farm, though it's at least straightforwardly benevolent, doesn't help people on the scale that Google does.

Most explicitly benevolent projects don't hold themselves sufficiently accountable. They act as if having good intentions were enough to guarantee good effects.

[3] Users dislike their new operating system so much that they're starting petitions to save the old one. And the old one was nothing special. The hackers within Microsoft must know in their hearts that if the company really cared about users they'd just advise them to switch to OSX.

Thanks to Trevor Blackwell, Paul Buchheit, Jessica Livingston, and Robert Morris for reading drafts of this.

- Russian Translation

- German Translation

Lies We Tell Kids

May 2008

Adults lie constantly to kids. I'm not saying we should stop, but I think we should at least examine which lies we tell and why.

There may also be a benefit to us. We were all lied to as kids, and some of the lies we were told still affect us. So by studying the ways adults lie to kids, we may be able to clear our heads of lies we were told.

I'm using the word "lie" in a very general sense: not just overt falsehoods, but also all the more subtle ways we mislead kids. Though "lie" has negative connotations, I don't mean to suggest we should never do this—just that we should pay attention when we do. [1]

One of the most remarkable things about the way we lie to kids is how broad the conspiracy is. All adults know what their culture lies to kids about: they're the questions you answer "Ask your parents." If a kid asked who won the World Series in 1982 or what the atomic weight of carbon was, you could just tell him. But if a kid asks you "Is there a God?" or "What's a prostitute?" you'll probably say "Ask your parents."

Since we all agree, kids see few cracks in the view of the world presented to them. The biggest disagreements are between parents and schools, but even those are small. Schools are careful what they say about controversial topics, and if they do contradict what parents want their kids to believe, parents either pressure the school into keeping quiet or move their kids to a new school.

The conspiracy is so thorough that most kids who discover it do so only by discovering internal contradictions in what they're told. It can be traumatic for the ones who wake up during the operation. Here's what happened to Einstein:

> Through the reading of popular scientific books I soon reached the conviction that much in the stories of the Bible could not be true. The consequence was a positively fanatic freethinking coupled with the impression that youth is intentionally being deceived by the state through lies: it was a crushing impression. [2]

I remember that feeling. By 15 I was convinced the world was corrupt from end to end. That's why movies like *The Matrix* have such resonance. Every kid grows up in

a fake world. In a way it would be easier if the forces behind it were as clearly differentiated as a bunch of evil machines, and one could make a clean break just by taking a pill.

Protection

If you ask adults why they lie to kids, the most common reason they give is to protect them. And kids do need protecting. The environment you want to create for a newborn child will be quite unlike the streets of a big city.

That seems so obvious it seems wrong to call it a lie. It's certainly not a bad lie to tell, to give a baby the impression the world is quiet and warm and safe. But this harmless type of lie can turn sour if left unexamined.

Imagine if you tried to keep someone in as protected an environment as a newborn till age 18. To mislead someone so grossly about the world would seem not protection but abuse. That's an extreme example, of course; when parents do that sort of thing it becomes national news. But you see the same problem on a smaller scale in the malaise teenagers feel in suburbia.

The main purpose of suburbia is to provide a protected environment for children to grow up in. And it seems great for 10 year olds. I liked living in suburbia when I was 10. I didn't notice how sterile it was. My whole world was no bigger than a few friends' houses I bicycled to and some woods I ran around in. On a log scale I was midway between crib and globe. A suburban street was just the right size. But as I grew older, suburbia started to feel suffocatingly fake.

Life can be pretty good at 10 or 20, but it's often frustrating at 15. This is too big a problem to solve here, but certainly one reason life sucks at 15 is that kids are trapped in a world designed for 10 year olds.

What do parents hope to protect their children from by raising them in suburbia? A friend who moved out of Manhattan said merely that her 3 year old daughter "saw too much." Off the top of my head, that might include: people who are high or drunk, poverty, madness, gruesome medical conditions, sexual behavior of various degrees of oddness, and violent anger.

I think it's the anger that would worry me most if I had a 3 year old. I was 29 when I moved to New York and I was surprised even then. I wouldn't want a 3 year old to see some of the disputes I saw. It would be too frightening. A lot of the things adults conceal from smaller children, they conceal because they'd be frightening, not because they want to conceal the existence of such things. Misleading the child is just a byproduct.

This seems one of the most justifiable types of lying adults do to kids. But because the lies are indirect we don't keep a very strict accounting of them. Parents know they've concealed the facts about sex, and many at some point sit their kids down and explain more. But few tell their kids about the differences between the real

world and the cocoon they grew up in. Combine this with the confidence parents try to instill in their kids, and every year you get a new crop of 18 year olds who think they know how to run the world.

Don't all 18 year olds think they know how to run the world? Actually this seems to be a recent innovation, no more than about 100 years old. In preindustrial times teenage kids were junior members of the adult world and comparatively well aware of their shortcomings. They could see they weren't as strong or skillful as the village smith. In past times people lied to kids about some things more than we do now, but the lies implicit in an artificial, protected environment are a recent invention. Like a lot of new inventions, the rich got this first. Children of kings and great magnates were the first to grow up out of touch with the world. Suburbia means half the population can live like kings in that respect.

Sex (and Drugs)

I'd have different worries about raising teenage kids in New York. I'd worry less about what they'd see, and more about what they'd do. I went to college with a lot of kids who grew up in Manhattan, and as a rule they seemed pretty jaded. They seemed to have lost their virginity at an average of about 14 and by college had tried more drugs than I'd even heard of.

The reasons parents don't want their teenage kids having sex are complex. There are some obvious dangers: pregnancy and sexually transmitted diseases. But those aren't the only reasons parents don't want their kids having sex. The average parents of a 14 year old girl would hate the idea of her having sex even if there were zero risk of pregnancy or sexually transmitted diseases.

Kids can probably sense they aren't being told the whole story. After all, pregnancy and sexually transmitted diseases are just as much a problem for adults, and they have sex.

What really bothers parents about their teenage kids having sex? Their dislike of the idea is so visceral it's probably inborn. But if it's inborn it should be universal, and there are plenty of societies where parents don't mind if their teenage kids have sex—indeed, where it's normal for 14 year olds to become mothers. So what's going on? There does seem to be a universal taboo against sex with prepubescent children. One can imagine evolutionary reasons for that. And I think this is the main reason parents in industrialized societies dislike teenage kids having sex. They still think of them as children, even though biologically they're not, so the taboo against child sex still has force.

One thing adults conceal about sex they also conceal about drugs: that it can cause great pleasure. That's what makes sex and drugs so dangerous. The desire for them can cloud one's judgement—which is especially frightening when the judgement being clouded is the already wretched judgement of a teenage kid.

Here parents' desires conflict. Older societies told kids they had bad judgement,

but modern parents want their children to be confident. This may well be a better plan than the old one of putting them in their place, but it has the side effect that after having implicitly lied to kids about how good their judgement is, we then have to lie again about all the things they might get into trouble with if they believed us.

If parents told their kids the truth about sex and drugs, it would be: the reason you should avoid these things is that you have lousy judgement. People with twice your experience still get burned by them. But this may be one of those cases where the truth wouldn't be convincing, because one of the symptoms of bad judgement is believing you have good judgement. When you're too weak to lift something, you can tell, but when you're making a decision impetuously, you're all the more sure of it.

Innocence

Another reason parents don't want their kids having sex is that they want to keep them innocent. Adults have a certain model of how kids are supposed to behave, and it's different from what they expect of other adults.

One of the most obvious differences is the words kids are allowed to use. Most parents use words when talking to other adults that they wouldn't want their kids using. They try to hide even the existence of these words for as long as they can. And this is another of those conspiracies everyone participates in: everyone knows you're not supposed to swear in front of kids.

I've never heard more different explanations for anything parents tell kids than why they shouldn't swear. Every parent I know forbids their children to swear, and yet no two of them have the same justification. It's clear most start with not wanting kids to swear, then make up the reason afterward.

So my theory about what's going on is that the *function* of swearwords is to mark the speaker as an adult. There's no difference in the meaning of "shit" and "poopoo." So why should one be ok for kids to say and one forbidden? The only explanation is: by definition. [3]

Why does it bother adults so much when kids do things reserved for adults? The idea of a foul-mouthed, cynical 10 year old leaning against a lamppost with a cigarette hanging out of the corner of his mouth is very disconcerting. But why?

One reason we want kids to be innocent is that we're programmed to like certain kinds of helplessness. I've several times heard mothers say they deliberately refrained from correcting their young children's mispronunciations because they were so cute. And if you think about it, cuteness is helplessness. Toys and cartoon characters meant to be cute always have clueless expressions and stubby, ineffectual limbs.

It's not surprising we'd have an inborn desire to love and protect helpless

creatures, considering human offspring are so helpless for so long. Without the helplessness that makes kids cute, they'd be very annoying. They'd merely seem like incompetent adults. But there's more to it than that. The reason our hypothetical jaded 10 year old bothers me so much is not just that he'd be annoying, but that he'd have cut off his prospects for growth so early. To be jaded you have to think you know how the world works, and any theory a 10 year old had about that would probably be a pretty narrow one.

Innocence is also open-mindedness. We want kids to be innocent so they can continue to learn. Paradoxical as it sounds, there are some kinds of knowledge that get in the way of other kinds of knowledge. If you're going to learn that the world is a brutal place full of people trying to take advantage of one another, you're better off learning it last. Otherwise you won't bother learning much more.

Very smart adults often seem unusually innocent, and I don't think this is a coincidence. I think they've deliberately avoided learning about certain things. Certainly I do. I used to think I wanted to know everything. Now I know I don't.

Death

After sex, death is the topic adults lie most conspicuously about to kids. Sex I believe they conceal because of deep taboos. But why do we conceal death from kids? Probably because small children are particularly horrified by it. They want to feel safe, and death is the ultimate threat.

One of the most spectacular lies our parents told us was about the death of our first cat. Over the years, as we asked for more details, they were compelled to invent more, so the story grew quite elaborate. The cat had died at the vet's office. Of what? Of the anaesthesia itself. Why was the cat at the vet's office? To be fixed. And why had such a routine operation killed it? It wasn't the vet's fault; the cat had a congenitally weak heart; the anaesthesia was too much for it; but there was no way anyone could have known this in advance. It was not till we were in our twenties that the truth came out: my sister, then about three, had accidentally stepped on the cat and broken its back.

They didn't feel the need to tell us the cat was now happily in cat heaven. My parents never claimed that people or animals who died had "gone to a better place," or that we'd meet them again. It didn't seem to harm us.

My grandmother told us an edited version of the death of my grandfather. She said they'd been sitting reading one day, and when she said something to him, he didn't answer. He seemed to be asleep, but when she tried to rouse him, she couldn't. "He was gone." Having a heart attack sounded like falling asleep. Later I learned it hadn't been so neat, and the heart attack had taken most of a day to kill him.

Along with such outright lies, there must have been a lot of changing the subject when death came up. I can't remember that, of course, but I can infer it from the fact that I didn't really grasp I was going to die till I was about 19. How could I

have missed something so obvious for so long? Now that I've seen parents managing the subject, I can see how: questions about death are gently but firmly turned aside.

On this topic, especially, they're met half-way by kids. Kids often want to be lied to. They want to believe they're living in a comfortable, safe world as much as their parents want them to believe it. [4]

Identity

Some parents feel a strong adherence to an ethnic or religious group and want their kids to feel it too. This usually requires two different kinds of lying: the first is to tell the child that he or she is an X, and the second is whatever specific lies Xes differentiate themselves by believing. [5]

Telling a child they have a particular ethnic or religious identity is one of the stickiest things you can tell them. Almost anything else you tell a kid, they can change their mind about later when they start to think for themselves. But if you tell a kid they're a member of a certain group, that seems nearly impossible to shake.

This despite the fact that it can be one of the most premeditated lies parents tell. When parents are of different religions, they'll often agree between themselves that their children will be "raised as Xes." And it works. The kids obligingly grow up considering themselves as Xes, despite the fact that if their parents had chosen the other way, they'd have grown up considering themselves as Ys.

One reason this works so well is the second kind of lie involved. The truth is common property. You can't distinguish your group by doing things that are rational, and believing things that are true. If you want to set yourself apart from other people, you have to do things that are arbitrary, and believe things that are false. And after having spent their whole lives doing things that are arbitrary and believing things that are false, and being regarded as odd by "outsiders" on that account, the cognitive dissonance pushing children to regard themselves as Xes must be enormous. If they aren't an X, why are they attached to all these arbitrary beliefs and customs? If they aren't an X, why do all the non-Xes call them one?

This form of lie is not without its uses. You can use it to carry a payload of beneficial beliefs, and they will also become part of the child's identity. You can tell the child that in addition to never wearing the color yellow, believing the world was created by a giant rabbit, and always snapping their fingers before eating fish, Xes are also particularly honest and industrious. Then X children will grow up feeling it's part of their identity to be honest and industrious.

This probably accounts for a lot of the spread of modern religions, and explains why their doctrines are a combination of the useful and the bizarre. The bizarre half is what makes the religion stick, and the useful half is the payload. [6]

Authority

One of the least excusable reasons adults lie to kids is to maintain power over them. Sometimes these lies are truly sinister, like a child molester telling his victims they'll get in trouble if they tell anyone what happened to them. Others seem more innocent; it depends how badly adults lie to maintain their power, and what they use it for.

Most adults make some effort to conceal their flaws from children. Usually their motives are mixed. For example, a father who has an affair generally conceals it from his children. His motive is partly that it would worry them, partly that this would introduce the topic of sex, and partly (a larger part than he would admit) that he doesn't want to tarnish himself in their eyes.

If you want to learn what lies are told to kids, read almost any book written to teach them about "issues." [7] Peter Mayle wrote one called *Why Are We Getting a Divorce?* It begins with the three most important things to remember about divorce, one of which is:

> You shouldn't put the blame on one parent, because divorce is never only one person's fault. [8]

Really? When a man runs off with his secretary, is it always partly his wife's fault? But I can see why Mayle might have said this. Maybe it's more important for kids to respect their parents than to know the truth about them.

But because adults conceal their flaws, and at the same time insist on high standards of behavior for kids, a lot of kids grow up feeling they fall hopelessly short. They walk around feeling horribly evil for having used a swearword, while in fact most of the adults around them are doing much worse things.

This happens in intellectual as well as moral questions. The more confident people are, the more willing they seem to be to answer a question "I don't know." Less confident people feel they have to have an answer or they'll look bad. My parents were pretty good about admitting when they didn't know things, but I must have been told a lot of lies of this type by teachers, because I rarely heard a teacher say "I don't know" till I got to college. I remember because it was so surprising to hear someone say that in front of a class.

The first hint I had that teachers weren't omniscient came in sixth grade, after my father contradicted something I'd learned in school. When I protested that the teacher had said the opposite, my father replied that the guy had no idea what he was talking about—that he was just an elementary school teacher, after all.

Just a teacher? The phrase seemed almost grammatically ill-formed. Didn't teachers know everything about the subjects they taught? And if not, why were they the ones teaching us?

The sad fact is, US public school teachers don't generally understand the stuff they're teaching very well. There are some sterling exceptions, but as a rule people planning to go into teaching rank academically near the bottom of the college population. So the fact that I still thought at age 11 that teachers were infallible shows what a job the system must have done on my brain.

School

What kids get taught in school is a complex mix of lies. The most excusable are those told to simplify ideas to make them easy to learn. The problem is, a lot of propaganda gets slipped into the curriculum in the name of simplification.

Public school textbooks represent a compromise between what various powerful groups want kids to be told. The lies are rarely overt. Usually they consist either of omissions or of over-emphasizing certain topics at the expense of others. The view of history we got in elementary school was a crude hagiography, with at least one representative of each powerful group.

The famous scientists I remember were Einstein, Marie Curie, and George Washington Carver. Einstein was a big deal because his work led to the atom bomb. Marie Curie was involved with X-rays. But I was mystified about Carver. He seemed to have done stuff with peanuts.

It's obvious now that he was on the list because he was black (and for that matter that Marie Curie was on it because she was a woman), but as a kid I was confused for years about him. I wonder if it wouldn't have been better just to tell us the truth: that there weren't any famous black scientists. Ranking George Washington Carver with Einstein misled us not only about science, but about the obstacles blacks faced in his time.

As subjects got softer, the lies got more frequent. By the time you got to politics and recent history, what we were taught was pretty much pure propaganda. For example, we were taught to regard political leaders as saints—especially the recently martyred Kennedy and King. It was astonishing to learn later that they'd both been serial womanizers, and that Kennedy was a speed freak to boot. (By the time King's plagiarism emerged, I'd lost the ability to be surprised by the misdeeds of famous people.)

I doubt you could teach kids recent history without teaching them lies, because practically everyone who has anything to say about it has some kind of spin to put on it. Much recent history *consists* of spin. It would probably be better just to teach them metafacts like that.

Probably the biggest lie told in schools, though, is that the way to succeed is through following "the rules." In fact most such rules are just hacks to manage large groups efficiently.

Peace

Of all the reasons we lie to kids, the most powerful is probably the same mundane reason they lie to us.

Often when we lie to people it's not part of any conscious strategy, but because they'd react violently to the truth. Kids, almost by definition, lack self-control. They react violently to things—and so they get lied to a lot. [9]

A few Thanksgivings ago, a friend of mine found himself in a situation that perfectly illustrates the complex motives we have when we lie to kids. As the roast turkey appeared on the table, his alarmingly perceptive 5 year old son suddenly asked if the turkey had wanted to die. Foreseeing disaster, my friend and his wife rapidly improvised: yes, the turkey had wanted to die, and in fact had lived its whole life with the aim of being their Thanksgiving dinner. And that (phew) was the end of that.

Whenever we lie to kids to protect them, we're usually also lying to keep the peace.

One consequence of this sort of calming lie is that we grow up thinking horrible things are normal. It's hard for us to feel a sense of urgency as adults over something we've literally been trained not to worry about. When I was about 10 I saw a documentary on pollution that put me into a panic. It seemed the planet was being irretrievably ruined. I went to my mother afterward to ask if this was so. I don't remember what she said, but she made me feel better, so I stopped worrying about it.

That was probably the best way to handle a frightened 10 year old. But we should understand the price. This sort of lie is one of the main reasons bad things persist: we're all trained to ignore them.

Detox

A sprinter in a race almost immediately enters a state called "oxygen debt." His body switches to an emergency source of energy that's faster than regular aerobic respiration. But this process builds up waste products that ultimately require extra oxygen to break down, so at the end of the race he has to stop and pant for a while to recover.

We arrive at adulthood with a kind of truth debt. We were told a lot of lies to get us (and our parents) through our childhood. Some may have been necessary. Some probably weren't. But we all arrive at adulthood with heads full of lies.

There's never a point where the adults sit you down and explain all the lies they told you. They've forgotten most of them. So if you're going to clear these lies out of your head, you're going to have to do it yourself.

Few do. Most people go through life with bits of packing material adhering to their

minds and never know it. You probably never can completely undo the effects of lies you were told as a kid, but it's worth trying. I've found that whenever I've been able to undo a lie I was told, a lot of other things fell into place.

Fortunately, once you arrive at adulthood you get a valuable new resource you can use to figure out what lies you were told. You're now one of the liars. You get to watch behind the scenes as adults spin the world for the next generation of kids.

The first step in clearing your head is to realize how far you are from a neutral observer. When I left high school I was, I thought, a complete skeptic. I'd realized high school was crap. I thought I was ready to question everything I knew. But among the many other things I was ignorant of was how much debris there already was in my head. It's not enough to consider your mind a blank slate. You have to consciously erase it.

Notes

[1] One reason I stuck with such a brutally simple word is that the lies we tell kids are probably not quite as harmless as we think. If you look at what adults told children in the past, it's shocking how much they lied to them. Like us, they did it with the best intentions. So if we think we're as open as one could reasonably be with children, we're probably fooling ourselves. Odds are people in 100 years will be as shocked at some of the lies we tell as we are at some of the lies people told 100 years ago.

I can't predict which these will be, and I don't want to write an essay that will seem dumb in 100 years. So instead of using special euphemisms for lies that seem excusable according to present fashions, I'm just going to call all our lies lies.

(I have omitted one type: lies told to play games with kids' credulity. These range from "make-believe," which is not really a lie because it's told with a wink, to the frightening lies told by older siblings. There's not much to say about these: I wouldn't want the first type to go away, and wouldn't expect the second type to.)

[2] Calaprice, Alice (ed.), *The Quotable Einstein*, Princeton University Press, 1996.

[3] If you ask parents why kids shouldn't swear, the less educated ones usually reply with some question-begging answer like "it's inappropriate," while the more educated ones come up with elaborate rationalizations. In fact the less educated parents seem closer to the truth.

[4] As a friend with small children pointed out, it's easy for small children to consider themselves immortal, because time seems to pass so slowly for them. To

a 3 year old, a day feels like a month might to an adult. So 80 years sounds to him like 2400 years would to us.

[5] I realize I'm going to get endless grief for classifying religion as a type of lie. Usually people skirt that issue with some equivocation implying that lies believed for a sufficiently long time by sufficiently large numbers of people are immune to the usual standards for truth. But because I can't predict which lies future generations will consider inexcusable, I can't safely omit any type we tell. Yes, it seems unlikely that religion will be out of fashion in 100 years, but no more unlikely than it would have seemed to someone in 1880 that schoolchildren in 1980 would be taught that masturbation was perfectly normal and not to feel guilty about it.

[6] Unfortunately the payload can consist of bad customs as well as good ones. For example, there are certain qualities that some groups in America consider "acting white." In fact most of them could as accurately be called "acting Japanese." There's nothing specifically white about such customs. They're common to all cultures with long traditions of living in cities. So it is probably a losing bet for a group to consider behaving the opposite way as part of its identity.

[7] In this context, "issues" basically means "things we're going to lie to them about." That's why there's a special name for these topics.

[8] Mayle, Peter, *Why Are We Getting a Divorce?*, Harmony, 1988.

[9] The ironic thing is, this is also the main reason kids lie to adults. If you freak out when people tell you alarming things, they won't tell you them. Teenagers don't tell their parents what happened that night they were supposed to be staying at a friend's house for the same reason parents don't tell 5 year olds the truth about the Thanksgiving turkey. They'd freak if they knew.

Thanks to Sam Altman, Marc Andreessen, Trevor Blackwell, Patrick Collison, Jessica Livingston, Jackie McDonough, Robert Morris, and David Sloo for reading drafts of this. And since there are some controversial ideas here, I should add that none of them agreed with everything in it.

- German Translation
- French Translation
- Russian Translation

Disconnecting Distraction

Note: The strategy described at the end of this essay didn't work. It would work for a while, and then I'd gradually find myself using the Internet on my work computer. I'm trying other strategies now, but I think this time I'll wait till I'm sure they work before writing about them.

May 2008

Procrastination feeds on distractions. Most people find it uncomfortable just to sit and do nothing; you avoid work by doing something else.

So one way to beat procrastination is to starve it of distractions. But that's not as straightforward as it sounds, because there are people working hard to distract you. Distraction is not a static obstacle that you avoid like you might avoid a rock in the road. Distraction seeks you out.

Chesterfield described dirt as matter out of place. Distracting is, similarly, desirable at the wrong time. And technology is continually being refined to produce more and more desirable things. Which means that as we learn to avoid one class of distractions, new ones constantly appear, like drug-resistant bacteria.

Television, for example, has after 50 years of refinement reached the point where it's like visual crack. I realized when I was 13 that TV was addictive, so I stopped watching it. But I read recently that the average American watches 4 hours of TV a day. A quarter of their life.

TV is in decline now, but only because people have found even more addictive ways of wasting time. And what's especially dangerous is that many happen at your computer. This is no accident. An ever larger percentage of office workers sit in front of computers connected to the Internet, and distractions always evolve toward the procrastinators.

I remember when computers were, for me at least, exclusively for work. I might occasionally dial up a server to get mail or ftp files, but most of the time I was offline. All I could do was write and program. Now I feel as if someone snuck a television onto my desk. Terribly addictive things are just a click away. Run into an obstacle in what you're working on? Hmm, I wonder what's new online. Better check.

After years of carefully avoiding classic time sinks like TV, games, and Usenet, I still managed to fall prey to distraction, because I didn't realize that it evolves. Something that used to be safe, using the Internet, gradually became more and more dangerous. Some days I'd wake up, get a cup of tea and check the news, then check email, then check the news again, then answer a few emails, then suddenly notice it was almost lunchtime and I hadn't gotten any real work done. And this started to happen more and more often.

It took me surprisingly long to realize how distracting the Internet had become, because the problem was intermittent. I ignored it the way you let yourself ignore a bug that only appears intermittently. When I was in the middle of a project, distractions weren't really a problem. It was when I'd finished one project and was deciding what to do next that they always bit me.

Another reason it was hard to notice the danger of this new type of distraction was that social customs hadn't yet caught up with it. If I'd spent a whole morning sitting on a sofa watching TV, I'd have noticed very quickly. That's a known danger sign, like drinking alone. But using the Internet still looked and felt a lot like work.

Eventually, though, it became clear that the Internet had become so much more distracting that I had to start treating it differently. Basically, I had to add a new application to my list of known time sinks: Firefox.

* * *

The problem is a hard one to solve because most people still need the Internet for some things. If you drink too much, you can solve that problem by stopping entirely. But you can't solve the problem of overeating by stopping eating. I couldn't simply avoid the Internet entirely, as I'd done with previous time sinks.

At first I tried rules. For example, I'd tell myself I was only going to use the Internet twice a day. But these schemes never worked for long. Eventually something would come up that required me to use it more than that. And then I'd gradually slip back into my old ways.

Addictive things have to be treated as if they were sentient adversaries—as if there were a little man in your head always cooking up the most plausible arguments for doing whatever you're trying to stop doing. If you leave a path to it, he'll find it.

The key seems to be visibility. The biggest ingredient in most bad habits is denial. So you have to make it so that you can't merely *slip* into doing the thing you're trying to avoid. It has to set off alarms.

Maybe in the long term the right answer for dealing with Internet distractions will be software that watches and controls them. But in the meantime I've found a more drastic solution that definitely works: to set up a separate computer for using the Internet.

I now leave wifi turned off on my main computer except when I need to transfer a file or edit a web page, and I have a separate laptop on the other side of the room that I use to check mail or browse the web. (Irony of ironies, it's the computer Steve Huffman wrote Reddit on. When Steve and Alexis auctioned off their old laptops for charity, I bought them for the Y Combinator museum.)

My rule is that I can spend as much time online as I want, as long as I do it on that computer. And this turns out to be enough. When I have to sit on the other side of the room to check email or browse the web, I become much more aware of it. Sufficiently aware, in my case at least, that it's hard to spend more than about an hour a day online.

And my main computer is now freed for work. If you try this trick, you'll probably be struck by how different it feels when your computer is disconnected from the Internet. It was alarming to me how foreign it felt to sit in front of a computer that could only be used for work, because that showed how much time I must have been wasting.

Wow. All I can do at this computer is work. Ok, I better work then.

That's the good part. Your old bad habits now help you to work. You're used to sitting in front of that computer for hours at a time. But you can't browse the web or check email now. What are you going to do? You can't just sit there. So you start working.

- Good and Bad Procrastination
- Spanish Translation
- Arabic Translation
- Catalan Translation
- Russian Translation
- Spanish Translation

Cities and Ambition

May 2008

Great cities attract ambitious people. You can sense it when you walk around one. In a hundred subtle ways, the city sends you a message: you could do more; you should try harder.

The surprising thing is how different these messages can be. New York tells you, above all: you should make more money. There are other messages too, of course. You should be hipper. You should be better looking. But the clearest message is that you should be richer.

What I like about Boston (or rather Cambridge) is that the message there is: you should be smarter. You really should get around to reading all those books you've been meaning to.

When you ask what message a city sends, you sometimes get surprising answers. As much as they respect brains in Silicon Valley, the message the Valley sends is: you should be more powerful.

That's not quite the same message New York sends. Power matters in New York too of course, but New York is pretty impressed by a billion dollars even if you merely inherited it. In Silicon Valley no one would care except a few real estate agents. What matters in Silicon Valley is how much effect you have on the world. The reason people there care about Larry and Sergey is not their wealth but the fact that they control Google, which affects practically everyone.

How much does it matter what message a city sends? Empirically, the answer seems to be: a lot. You might think that if you had enough strength of mind to do great things, you'd be able to transcend your environment. Where you live should make at most a couple percent difference. But if you look at the historical evidence, it seems to matter more than that. Most people who did great things were clumped together in a few places where that sort of thing was done at the time.

You can see how powerful cities are from something I wrote about earlier: the case

of the Milanese Leonardo. Practically every fifteenth century Italian painter you've heard of was from Florence, even though Milan was just as big. People in Florence weren't genetically different, so you have to assume there was someone born in Milan with as much natural ability as Leonardo. What happened to him?

If even someone with the same natural ability as Leonardo couldn't beat the force of environment, do you suppose you can?

I don't. I'm fairly stubborn, but I wouldn't try to fight this force. I'd rather use it. So I've thought a lot about where to live.

I'd always imagined Berkeley would be the ideal place — that it would basically be Cambridge with good weather. But when I finally tried living there a couple years ago, it turned out not to be. The message Berkeley sends is: you should live better. Life in Berkeley is very civilized. It's probably the place in America where someone from Northern Europe would feel most at home. But it's not humming with ambition.

In retrospect it shouldn't have been surprising that a place so pleasant would attract people interested above all in quality of life. Cambridge with good weather, it turns out, is not Cambridge. The people you find in Cambridge are not there by accident. You have to make sacrifices to live there. It's expensive and somewhat grubby, and the weather's often bad. So the kind of people you find in Cambridge are the kind of people who want to live where the smartest people are, even if that means living in an expensive, grubby place with bad weather.

As of this writing, Cambridge seems to be the intellectual capital of the world. I realize that seems a preposterous claim. What makes it true is that it's more preposterous to claim about anywhere else. American universities currently seem to be the best, judging from the flow of ambitious students. And what US city has a stronger claim? New York? A fair number of smart people, but diluted by a much larger number of neanderthals in suits. The Bay Area has a lot of smart people too, but again, diluted; there are two great universities, but they're far apart. Harvard and MIT are practically adjacent by West Coast standards, and they're surrounded by about 20 other colleges and universities. [1]

Cambridge as a result feels like a town whose main industry is ideas, while New York's is finance and Silicon Valley's is startups.

When you talk about cities in the sense we are, what you're really talking about is collections of people. For a long time cities were the only large collections of people, so you could use the two ideas interchangeably. But we can see how much things are changing from the examples I've mentioned. New York is a classic great city. But Cambridge is just part of a city, and Silicon Valley is not even that. (San Jose is not, as it sometimes claims, the capital of Silicon Valley. It's just 178

square miles at one end of it.)

Maybe the Internet will change things further. Maybe one day the most important community you belong to will be a virtual one, and it won't matter where you live physically. But I wouldn't bet on it. The physical world is very high bandwidth, and some of the ways cities send you messages are quite subtle.

One of the exhilarating things about coming back to Cambridge every spring is walking through the streets at dusk, when you can see into the houses. When you walk through Palo Alto in the evening, you see nothing but the blue glow of TVs. In Cambridge you see shelves full of promising-looking books. Palo Alto was probably much like Cambridge in 1960, but you'd never guess now that there was a university nearby. Now it's just one of the richer neighborhoods in Silicon Valley. [2]

A city speaks to you mostly by accident — in things you see through windows, in conversations you overhear. It's not something you have to seek out, but something you can't turn off. One of the occupational hazards of living in Cambridge is overhearing the conversations of people who use interrogative intonation in declarative sentences. But on average I'll take Cambridge conversations over New York or Silicon Valley ones.

A friend who moved to Silicon Valley in the late 90s said the worst thing about living there was the low quality of the eavesdropping. At the time I thought she was being deliberately eccentric. Sure, it can be interesting to eavesdrop on people, but is good quality eavesdropping so important that it would affect where you chose to live? Now I understand what she meant. The conversations you overhear tell you what sort of people you're among.

No matter how determined you are, it's hard not to be influenced by the people around you. It's not so much that you do whatever a city expects of you, but that you get discouraged when no one around you cares about the same things you do.

There's an imbalance between encouragement and discouragement like that between gaining and losing money. Most people overvalue negative amounts of money: they'll work much harder to avoid losing a dollar than to gain one. Similarly, although there are plenty of people strong enough to resist doing something just because that's what one is supposed to do where they happen to be, there are few strong enough to keep working on something no one around them cares about.

Because ambitions are to some extent incompatible and admiration is a zero-sum game, each city tends to focus on one type of ambition. The reason Cambridge is the intellectual capital is not just that there's a concentration of smart people there, but that there's nothing *else* people there care about more. Professors in

New York and the Bay area are second class citizens — till they start hedge funds or startups respectively.

This suggests an answer to a question people in New York have wondered about since the Bubble: whether New York could grow into a startup hub to rival Silicon Valley. One reason that's unlikely is that someone starting a startup in New York would feel like a second class citizen. [3] There's already something else people in New York admire more.

In the long term, that could be a bad thing for New York. The power of an important new technology does eventually convert to money. So by caring more about money and less about power than Silicon Valley, New York is recognizing the same thing, but slower. [4] And in fact it has been losing to Silicon Valley at its own game: the ratio of New York to California residents in the Forbes 400 has decreased from 1.45 (81:56) when the list was first published in 1982 to .83 (73:88) in 2007.

Not all cities send a message. Only those that are centers for some type of ambition do. And it can be hard to tell exactly what message a city sends without living there. I understand the messages of New York, Cambridge, and Silicon Valley because I've lived for several years in each of them. DC and LA seem to send messages too, but I haven't spent long enough in either to say for sure what they are.

The big thing in LA seems to be fame. There's an A List of people who are most in demand right now, and what's most admired is to be on it, or friends with those who are. Beneath that, the message is much like New York's, though perhaps with more emphasis on physical attractiveness.

In DC the message seems to be that the most important thing is who you know. You want to be an insider. In practice this seems to work much as in LA. There's an A List and you want to be on it or close to those who are. The only difference is how the A List is selected. And even that is not that different.

At the moment, San Francisco's message seems to be the same as Berkeley's: you should live better. But this will change if enough startups choose SF over the Valley. During the Bubble that was a predictor of failure — a self-indulgent choice, like buying expensive office furniture. Even now I'm suspicious when startups choose SF. But if enough good ones do, it stops being a self-indulgent choice, because the center of gravity of Silicon Valley will shift there.

I haven't found anything like Cambridge for intellectual ambition. Oxford and Cambridge (England) feel like Ithaca or Hanover: the message is there, but not as strong.

Paris was once a great intellectual center. If you went there in 1300, it might have sent the message Cambridge does now. But I tried living there for a bit last year, and the ambitions of the inhabitants are not intellectual ones. The message Paris sends now is: do things with style. I liked that, actually. Paris is the only city I've lived in where people genuinely cared about art. In America only a few rich people buy original art, and even the more sophisticated ones rarely get past judging it by the brand name of the artist. But looking through windows at dusk in Paris you can see that people there actually care what paintings look like. Visually, Paris has the best eavesdropping I know. [5]

There's one more message I've heard from cities: in London you can still (barely) hear the message that one should be more aristocratic. If you listen for it you can also hear it in Paris, New York, and Boston. But this message is everywhere very faint. It would have been strong 100 years ago, but now I probably wouldn't have picked it up at all if I hadn't deliberately tuned in to that wavelength to see if there was any signal left.

So far the complete list of messages I've picked up from cities is: wealth, style, hipness, physical attractiveness, fame, political power, economic power, intelligence, social class, and quality of life.

My immediate reaction to this list is that it makes me slightly queasy. I'd always considered ambition a good thing, but I realize now that was because I'd always implicitly understood it to mean ambition in the areas I cared about. When you list everything ambitious people are ambitious about, it's not so pretty.

On closer examination I see a couple things on the list that are surprising in the light of history. For example, physical attractiveness wouldn't have been there 100 years ago (though it might have been 2400 years ago). It has always mattered for women, but in the late twentieth century it seems to have started to matter for men as well. I'm not sure why — probably some combination of the increasing power of women, the increasing influence of actors as models, and the fact that so many people work in offices now: you can't show off by wearing clothes too fancy to wear in a factory, so you have to show off with your body instead.

Hipness is another thing you wouldn't have seen on the list 100 years ago. Or wouldn't you? What it means is to know what's what. So maybe it has simply replaced the component of social class that consisted of being "au fait." That could explain why hipness seems particularly admired in London: it's version 2 of the traditional English delight in obscure codes that only insiders understand.

Economic power would have been on the list 100 years ago, but what we mean by it is changing. It used to mean the control of vast human and material resources. But increasingly it means the ability to direct the course of technology, and some of the people in a position to do that are not even rich — leaders of important open

source projects, for example. The Captains of Industry of times past had laboratories full of clever people cooking up new technologies for them. The new breed are themselves those people.

As this force gets more attention, another is dropping off the list: social class. I think the two changes are related. Economic power, wealth, and social class are just names for the same thing at different stages in its life: economic power converts to wealth, and wealth to social class. So the focus of admiration is simply shifting upstream.

Does anyone who wants to do great work have to live in a great city? No; all great cities inspire some sort of ambition, but they aren't the only places that do. For some kinds of work, all you need is a handful of talented colleagues.

What cities provide is an audience, and a funnel for peers. These aren't so critical in something like math or physics, where no audience matters except your peers, and judging ability is sufficiently straightforward that hiring and admissions committees can do it reliably. In a field like math or physics all you need is a department with the right colleagues in it. It could be anywhere — in Los Alamos, New Mexico, for example.

It's in fields like the arts or writing or technology that the larger environment matters. In these the best practitioners aren't conveniently collected in a few top university departments and research labs — partly because talent is harder to judge, and partly because people pay for these things, so one doesn't need to rely on teaching or research funding to support oneself. It's in these more chaotic fields that it helps most to be in a great city: you need the encouragement of feeling that people around you care about the kind of work you do, and since you have to find peers for yourself, you need the much larger intake mechanism of a great city.

You don't have to live in a great city your whole life to benefit from it. The critical years seem to be the early and middle ones of your career. Clearly you don't have to grow up in a great city. Nor does it seem to matter if you go to college in one. To most college students a world of a few thousand people seems big enough. Plus in college you don't yet have to face the hardest kind of work — discovering new problems to solve.

It's when you move on to the next and much harder step that it helps most to be in a place where you can find peers and encouragement. You seem to be able to leave, if you want, once you've found both. The Impressionists show the typical pattern: they were born all over France (Pissarro was born in the Carribbean) and died all over France, but what defined them were the years they spent together in Paris.

Unless you're sure what you want to do and where the leading center for it is, your best bet is probably to try living in several places when you're young. You can never tell what message a city sends till you live there, or even whether it still sends one. Often your information will be wrong: I tried living in Florence when I was 25, thinking it would be an art center, but it turned out I was 450 years too late.

Even when a city is still a live center of ambition, you won't know for sure whether its message will resonate with you till you hear it. When I moved to New York, I was very excited at first. It's an exciting place. So it took me quite a while to realize I just wasn't like the people there. I kept searching for the Cambridge of New York. It turned out it was way, way uptown: an hour uptown by air.

Some people know at 16 what sort of work they're going to do, but in most ambitious kids, ambition seems to precede anything specific to be ambitious about. They know they want to do something great. They just haven't decided yet whether they're going to be a rock star or a brain surgeon. There's nothing wrong with that. But it means if you have this most common type of ambition, you'll probably have to figure out where to live by trial and error. You'll probably have to find the city where you feel at home to know what sort of ambition you have.

Notes

[1] This is one of the advantages of not having the universities in your country controlled by the government. When governments decide how to allocate resources, political deal-making causes things to be spread out geographically. No central goverment would put its two best universities in the same town, unless it was the capital (which would cause other problems). But scholars seem to like to cluster together as much as people in any other field, and when given the freedom to they derive the same advantages from it.

[2] There are still a few old professors in Palo Alto, but one by one they die and their houses are transformed by developers into McMansions and sold to VPs of Bus Dev.

[3] How many times have you read about startup founders who continued to live inexpensively as their companies took off? Who continued to dress in jeans and t-shirts, to drive the old car they had in grad school, and so on? If you did that in New York, people would treat you like shit. If you walk into a fancy restaurant in San Francisco wearing a jeans and a t-shirt, they're nice to you; who knows who

you might be? Not in New York.

One sign of a city's potential as a technology center is the number of restaurants that still require jackets for men. According to Zagat's there are none in San Francisco, LA, Boston, or Seattle, 4 in DC, 6 in Chicago, 8 in London, 13 in New York, and 20 in Paris.

(Zagat's lists the Ritz Carlton Dining Room in SF as requiring jackets but I couldn't believe it, so I called to check and in fact they don't. Apparently there's only one restaurant left on the entire West Coast that still requires jackets: The French Laundry in Napa Valley.)

[4] Ideas are one step upstream from economic power, so it's conceivable that intellectual centers like Cambridge will one day have an edge over Silicon Valley like the one the Valley has over New York.

This seems unlikely at the moment; if anything Boston is falling further and further behind. The only reason I even mention the possibility is that the path from ideas to startups has recently been getting smoother. It's a lot easier now for a couple of hackers with no business experience to start a startup than it was 10 years ago. If you extrapolate another 20 years, maybe the balance of power will start to shift back. I wouldn't bet on it, but I wouldn't bet against it either.

[5] If Paris is where people care most about art, why is New York the center of gravity of the art business? Because in the twentieth century, art as brand split apart from art as stuff. New York is where the richest buyers are, but all they demand from art is brand, and since you can base brand on anything with a sufficiently identifiable style, you may as well use the local stuff.

Thanks to Trevor Blackwell, Sarah Harlin, Jessica Livingston, Jackie McDonough, Robert Morris, and David Sloo for reading drafts of this.

- Italian Translation
- Portuguese Translation
- Chinese Translation
- Korean Translation

The Pooled-Risk Company Management Company

July 2008

At this year's startup school, David Heinemeier Hansson gave a talk in which he suggested that startup founders should do things the old fashioned way. Instead of hoping to get rich by building a valuable company and then selling stock in a "liquidity event," founders should start companies that make money and live off the revenues.

Sounds like a good plan. Let's think about the optimal way to do this.

One disadvantage of living off the revenues of your company is that you have to keep running it. And as anyone who runs their own business can tell you, that requires your complete attention. You can't just start a business and check out once things are going well, or they stop going well surprisingly fast.

The main economic motives of startup founders seem to be freedom and security. They want enough money that (a) they don't have to worry about running out of money and (b) they can spend their time how they want. Running your own business offers neither. You certainly don't have freedom: no boss is so demanding. Nor do you have security, because if you stop paying attention to the company, its revenues go away, and with them your income.

The best case, for most people, would be if you could hire someone to manage the company for you once you'd grown it to a certain size. Suppose you could find a really good manager. Then you would have both freedom and security. You could pay as little attention to the business as you wanted, knowing that your manager would keep things running smoothly. And that being so, revenues would continue to flow in, so you'd have security as well.

There will of course be some founders who wouldn't like that idea: the ones who like running their company so much that there's nothing else they'd rather do. But this group must be small. The way you succeed in most businesses is to be fanatically attentive to customers' needs. What are the odds that your own desires would coincide exactly with the demands of this powerful, external force?

Sure, running your own company can be fairly interesting. Viaweb was more

interesting than any job I'd had before. And since I made much more money from it, it offered the highest ratio of income to boringness of anything I'd done, by orders of magnitude. But was it *the* most interesting work I could imagine doing? No.

Whether the number of founders in the same position is asymptotic or merely large, there are certainly a lot of them. For them the right approach would be to hand the company over to a professional manager eventually, if they could find one who was good enough.

So far so good. But what if your manager was hit by a bus? What you really want is a management company to run your company for you. Then you don't depend on any one person.

If you own rental property, there are companies you can hire to manage it for you. Some will do everything, from finding tenants to fixing leaks. Of course, running companies is a lot more complicated than managing rental property, but let's suppose there were management companies that could do it for you. They'd charge a lot, but wouldn't it be worth it? I'd sacrifice a large percentage of the income for the extra peace of mind.

I realize what I'm describing already sounds too good to be true, but I can think of a way to make it even more attractive. If company management companies existed, there would be an additional service they could offer clients: they could let them insure their returns by pooling their risk. After all, even a perfect manager can't save a company when, as sometimes happens, its whole market dies, just as property managers can't save you from the building burning down. But a company that managed a large enough number of companies could say to all its clients: we'll combine the revenues from all your companies, and pay you your proportionate share.

If such management companies existed, they'd offer the maximum of freedom and security. Someone would run your company for you, and you'd be protected even if it happened to die.

Let's think about how such a management company might be organized. The simplest way would be to have a new kind of stock representing the total pool of companies they were managing. When you signed up, you'd trade your company's stock for shares of this pool, in proportion to an estimate of your company's value that you'd both agreed upon. Then you'd automatically get your share of the returns of the whole pool.

The catch is that because this kind of trade would be hard to undo, you couldn't switch management companies. But there's a way they could fix that: suppose all the company management companies got together and agreed to allow their

clients to exchange shares in all their pools. Then you could, in effect, simultaneously choose all the management companies to run yours for you, in whatever proportion you wanted, and change your mind later as often as you wanted.

If such pooled-risk company management companies existed, signing up with one would seem the ideal plan for most people following the route David advocated.

Good news: they do exist. What I've just described is an acquisition by a public company.

Unfortunately, though public acquirers are structurally identical to pooled-risk company management companies, they don't think of themselves that way. With a property management company, you can just walk in whenever you want and say "manage my rental property for me" and they'll do it. Whereas acquirers are, as of this writing, extremely fickle. Sometimes they're in a buying mood and they'll overpay enormously; other times they're not interested. They're like property management companies run by madmen. Or more precisely, by Benjamin Graham's Mr. Market.

So while on average public acquirers behave like pooled-risk company managers, you need a window of several years to get average case performance. If you wait long enough (five years, say) you're likely to hit an up cycle where some acquirer is hot to buy you. But you can't choose when it happens.

You can't assume investors will carry you for as long as you might have to wait. Your company has to make money. Opinions are divided about how early to focus on that. Joe Kraus says you should try charging customers right away. And yet some of the most successful startups, including Google, ignored revenue at first and concentrated exclusively on development. The answer probably depends on the type of company you're starting. I can imagine some where trying to make sales would be a good heuristic for product design, and others where it would just be a distraction. The test is probably whether it helps you to understand your users.

You can choose whichever revenue strategy you think is best for the type of company you're starting, so long as you're profitable. Being profitable ensures you'll get at least the average of the acquisition market—in which public companies do behave as pooled-risk company management companies.

David isn't mistaken in saying you should start a company to live off its revenues. The mistake is thinking this is somehow opposed to starting a company and selling it. In fact, for most people the latter is merely the optimal case of the former.

Thanks to Trevor Blackwell, Jessica Livingston, Michael Mandel, Robert Morris, and Fred Wilson for reading drafts of this.

- Russian Translation

A Fundraising Survival Guide

August 2008

Raising money is the second hardest part of starting a startup. The hardest part is making something people want: most startups that die, die because they didn't do that. But the second biggest cause of death is probably the difficulty of raising money. Fundraising is brutal.

One reason it's so brutal is simply the brutality of markets. People who've spent most of their lives in schools or big companies may not have been exposed to that. Professors and bosses usually feel some sense of responsibility toward you; if you make a valiant effort and fail, they'll cut you a break. Markets are less forgiving. Customers don't care how hard you worked, only whether you solved their problems.

Investors evaluate startups the way customers evaluate products, not the way bosses evaluate employees. If you're making a valiant effort and failing, maybe they'll invest in your next startup, but not this one.

But raising money from investors is harder than selling to customers, because there are so few of them. There's nothing like an efficient market. You're unlikely to have more than 10 who are interested; it's difficult to talk to more. So the randomness of any one investor's behavior can really affect you.

Problem number 3: investors are very random. All investors, including us, are by ordinary standards incompetent. We constantly have to make decisions about things we don't understand, and more often than not we're wrong.

And yet a lot is at stake. The amounts invested by different types of investors vary from five thousand dollars to fifty million, but the amount usually seems large for whatever type of investor it is. Investment decisions are big decisions.

That combination—making big decisions about things they don't understand—tends to make investors very skittish. VCs are notorious for leading founders on. Some of the more unscrupulous do it deliberately. But even the most well-intentioned investors can behave in a way that would seem crazy in everyday life. One day they're full of enthusiasm and seem ready to write you a check on the spot; the next they won't return your phone calls. They're not playing games with you. They just can't make up their minds. [1]

If that weren't bad enough, these wildly fluctuating nodes are all linked together. Startup investors all know one another, and (though they hate to admit it) the biggest factor in their opinion of you is the opinion of other investors. [2] Talk

about a recipe for an unstable system. You get the opposite of the damping that the fear/greed balance usually produces in markets. No one is interested in a startup that's a "bargain" because everyone else hates it.

So the inefficient market you get because there are so few players is exacerbated by the fact that they act less than independently. The result is a system like some kind of primitive, multi-celled sea creature, where you irritate one extremity and the whole thing contracts violently.

Y Combinator is working to fix this. We're trying to increase the number of investors just as we're increasing the number of startups. We hope that as the number of both increases we'll get something more like an efficient market. As t approaches infinity, Demo Day approaches an auction.

Unfortunately, t is still very far from infinity. What does a startup do now, in the imperfect world we currently inhabit? The most important thing is not to let fundraising get you down. Startups live or die on morale. If you let the difficulty of raising money destroy your morale, it will become a self-fulfilling prophecy.

Bootstrapping (= Consulting)

Some would-be founders may by now be thinking, why deal with investors at all? If raising money is so painful, why do it?

One answer to that is obvious: because you need money to live on. It's a fine idea in principle to finance your startup with its own revenues, but you can't create instant customers. Whatever you make, you have to sell a certain amount to break even. It will take time to grow your sales to that point, and it's hard to predict, till you try, how long it will take.

We could not have bootstrapped Viaweb, for example. We charged quite a lot for our software—about $140 per user per month—but it was at least a year before our revenues would have covered even our paltry costs. We didn't have enough saved to live on for a year.

If you factor out the "bootstrapped" companies that were actually funded by their founders through savings or a day job, the remainder either (a) got really lucky, which is hard to do on demand, or (b) began life as consulting companies and gradually transformed themselves into product companies.

Consulting is the only option you can count on. But consulting is far from free money. It's not as painful as raising money from investors, perhaps, but the pain is spread over a longer period. Years, probably. And for many types of startup, that delay could be fatal. If you're working on something so unusual that no one else is likely to think of it, you can take your time. Joshua Schachter gradually built Delicious on the side while working on Wall Street. He got away with it because no one else realized it was a good idea. But if you were building something as obviously necessary as online store software at about the same time as Viaweb, and you were working on it on the side while spending most of your time on client work, you were not in a good position.

Bootstrapping sounds great in principle, but this apparently verdant territory is one from which few startups emerge alive. The mere fact that bootstrapped startups tend to be famous on that account should set off alarm bells. If it worked so well, it

would be the norm. [3]

Bootstrapping may get easier, because starting a company is getting cheaper. But I don't think we'll ever reach the point where most startups can do without outside funding. Technology tends to get dramatically cheaper, but living expenses don't.

The upshot is, you can choose your pain: either the short, sharp pain of raising money, or the chronic ache of consulting. For a given total amount of pain, raising money is the better choice, because new technology is usually more valuable now than later.

But although for most startups raising money will be the lesser evil, it's still a pretty big evil—so big that it can easily kill you. Not merely in the obvious sense that if you fail to raise money you might have to shut the company down, but because the *process* of raising money itself can kill you.

To survive it you need a set of techniques mostly orthogonal to the ones used in convincing investors, just as mountain climbers need to know survival techniques that are mostly orthogonal to those used in physically getting up and down mountains.

1. Have low expectations.

The reason raising money destroys so many startups' morale is not simply that it's hard, but that it's so much harder than they expected. What kills you is the disappointment. And the lower your expectations, the harder it is to be disappointed.

Startup founders tend to be optimistic. This can work well in technology, at least some of the time, but it's the wrong way to approach raising money. Better to assume investors will always let you down. Acquirers too, while we're at it. At YC one of our secondary mantras is "Deals fall through." No matter what deal you have going on, assume it will fall through. The predictive power of this simple rule is amazing.

There will be a tendency, as a deal progresses, to start to believe it will happen, and then to depend on it happening. You must resist this. Tie yourself to the mast. This is what kills you. Deals do not have a trajectory like most other human interactions, where shared plans solidify linearly over time. Deals often fall through at the last moment. Often the other party doesn't really think about what they want till the last moment. So you can't use your everyday intuitions about shared plans as a guide. When it comes to deals, you have to consciously turn them off and become pathologically cynical.

This is harder to do than it sounds. It's very flattering when eminent investors seem interested in funding you. It's easy to start to believe that raising money will be quick and straightforward. But it hardly ever is.

2. Keep working on your startup.

It sounds obvious to say that you should keep working on your startup while raising money. Actually this is hard to do. Most startups don't manage to.

Raising money has a mysterious capacity to suck up all your attention. Even if you

only have one meeting a day with investors, somehow that one meeting will burn up your whole day. It costs not just the time of the actual meeting, but the time getting there and back, and the time preparing for it beforehand and thinking about it afterward.

The best way to survive the distraction of meeting with investors is probably to partition the company: to pick one founder to deal with investors while the others keep the company going. This works better when a startup has 3 founders than 2, and better when the leader of the company is not also the lead developer. In the best case, the company keeps moving forward at about half speed.

That's the best case, though. More often than not the company comes to a standstill while raising money. And that is dangerous for so many reasons. Raising money always takes longer than you expect. What seems like it's going to be a 2 week interruption turns into a 4 month interruption. That can be very demoralizing. And worse still, it can make you less attractive to investors. They want to invest in companies that are dynamic. A company that hasn't done anything new in 4 months doesn't seem dynamic, so they start to lose interest. Investors rarely grasp this, but much of what they're responding to when they lose interest in a startup is the damage done by their own indecision.

The solution: put the startup first. Fit meetings with investors into the spare moments in your development schedule, rather than doing development in the spare moments between meetings with investors. If you keep the company moving forward—releasing new features, increasing traffic, doing deals, getting written about—those investor meetings are more likely to be productive. Not just because your startup will seem more alive, but also because it will be better for your own morale, which is one of the main ways investors judge you.

3. Be conservative.

As conditions get worse, the optimal strategy becomes more conservative. When things go well you can take risks; when things are bad you want to play it safe.

I advise approaching fundraising as if it were always going badly. The reason is that between your ability to delude yourself and the wildly unstable nature of the system you're dealing with, things probably either already are or could easily become much worse than they seem.

What I tell most startups we fund is that if someone reputable offers you funding on reasonable terms, take it. There have been startups that ignored this advice and got away with it—startups that ignored a good offer in the hope of getting a better one, and actually did. But in the same position I'd give the same advice again. Who knows how many bullets were in the gun they were playing Russian roulette with?

Corollary: if an investor seems interested, don't just let them sit. You can't assume someone interested in investing will stay interested. In fact, you can't even tell (*they* can't even tell) if they're really interested till you try to convert that interest into money. So if you have hot prospect, either close them now or write them off. And unless you already have enough funding, that reduces to: close them now.

Startups don't win by getting great funding rounds, but by making great products. So finish raising money and get back to work.

4. Be flexible.

There are two questions VCs ask that you shouldn't answer: "Who else are you talking to?" and "How much are you trying to raise?"

VCs don't expect you to answer the first question. They ask it just in case. [4] They do seem to expect an answer to the second. But I don't think you should just tell them a number. Not as a way to play games with them, but because you shouldn't *have* a fixed amount you need to raise.

The custom of a startup needing a fixed amount of funding is an obsolete one left over from the days when startups were more expensive. A company that needed to build a factory or hire 50 people obviously needed to raise a certain minimum amount. But few technology startups are in that position today.

We advise startups to tell investors there are several different routes they could take depending on how much they raised. As little as $50k could pay for food and rent for the founders for a year. A couple hundred thousand would let them get office space and hire some smart people they know from school. A couple million would let them really blow this thing out. The message (and not just the message, but the fact) should be: we're going to succeed no matter what. Raising more money just lets us do it faster.

If you're raising an angel round, the size of the round can even change on the fly. In fact, it's just as well to make the round small initially, then expand as needed, rather than trying to raise a large round and risk losing the investors you already have if you can't raise the full amount. You may even want to do a "rolling close," where the round has no predetermined size, but instead you sell stock to investors one at a time as they say yes. That helps break deadlocks, because you can start as soon as the first one is ready to buy. [5]

5. Be independent.

A startup with a couple founders in their early twenties can have expenses so low that they could be profitable on as little as $2000 per month. That's negligible as corporate revenues go, but the effect on your morale and your bargaining position is anything but. At YC we use the phrase "ramen profitable" to describe the situation where you're making just enough to pay your living expenses. Once you cross into ramen profitable, everything changes. You may still need investment to make it big, but you don't need it this month.

You can't plan when you start a startup how long it will take to become profitable. But if you find yourself in a position where a little more effort expended on sales would carry you over the threshold of ramen profitable, do it.

Investors like it when you're ramen profitable. It shows you've thought about making money, instead of just working on amusing technical problems; it shows you have the discipline to keep your expenses low; but above all, it means you don't need them.

There is nothing investors like more than a startup that seems like it's going to succeed even without them. Investors like it when they can help a startup, but they don't like startups that would die without that help.

At YC we spend a lot of time trying to predict how the startups we've funded will do, because we're trying to learn how to pick winners. We've now watched the trajectories of so many startups that we're getting better at predicting them. And when we're talking about startups we think are likely to succeed, what we find ourselves saying is things like "Oh, those guys can take care of themselves. They'll be fine." Not "those guys are really smart" or "those guys are working on a great idea." [6] When we predict good outcomes for startups, the qualities that come up in the supporting arguments are toughness, adaptability, determination. Which means to the extent we're correct, those are the qualities you need to win.

Investors know this, at least unconsciously. The reason they like it when you don't need them is not simply that they like what they can't have, but because that quality is what makes founders succeed.

Sam Altman has it. You could parachute him into an island full of cannibals and come back in 5 years and he'd be the king. If you're Sam Altman, you don't have to be profitable to convey to investors that you'll succeed with or without them. (He wasn't, and he did.) Not everyone has Sam's deal-making ability. I myself don't. But if you don't, you can let the numbers speak for you.

6. Don't take rejection personally.

Getting rejected by investors can make you start to doubt yourself. After all, they're more experienced than you. If they think your startup is lame, aren't they probably right?

Maybe, maybe not. The way to handle rejection is with precision. You shouldn't simply ignore rejection. It might mean something. But you shouldn't automatically get demoralized either.

To understand what rejection means, you have to understand first of all how common it is. Statistically, the average VC is a rejection machine. David Hornik, a partner at August, told me:

> The numbers for me ended up being something like 500 to 800 plans received and read, somewhere between 50 and 100 initial 1 hour meetings held, about 20 companies that I got interested in, about 5 that I got serious about and did a bunch of work, 1 to 2 deals done in a year. So the odds are against you. You may be a great entrepreneur, working on interesting stuff, etc. but it is still incredibly unlikely that you get funded.

This is less true with angels, but VCs reject practically everyone. The structure of their business means a partner does at most 2 new investments a year, no matter how many good startups approach him.

In addition to the odds being terrible, the average investor is, as I mentioned, a pretty bad judge of startups. It's harder to judge startups than most other things, because great startup ideas tend to seem wrong. A good startup idea has to be not just good but novel. And to be both good and novel, an idea probably has to seem bad to most people, or someone would already be doing it and it wouldn't be novel.

That makes judging startups harder than most other things one judges. You have to be an intellectual contrarian to be a good startup investor. That's a problem for VCs, most of whom are not particularly imaginative. VCs are mostly money guys, not people who make things. [7] Angels are better at appreciating novel ideas, because most were founders themselves.

So when you get a rejection, use the data that's in it, and not what's not. If an investor gives you specific reasons for not investing, look at your startup and ask if they're right. If they're real problems, fix them. But don't just take their word for it. You're supposed to be the domain expert; you have to decide.

Though a rejection doesn't necessarily tell you anything about your startup, it does suggest your pitch could be improved. Figure out what's not working and change it. Don't just think "investors are stupid." Often they are, but figure out precisely where you lose them.

Don't let rejections pile up as a depressing, undifferentiated heap. Sort them and analyze them, and then instead of thinking "no one likes us," you'll know precisely how big a problem you have, and what to do about it.

7. Be able to downshift into consulting (if appropriate).

Consulting, as I mentioned, is a dangerous way to finance a startup. But it's better than dying. It's a bit like anaerobic respiration: not the optimum solution for the long term, but it can save you from an immediate threat. If you're having trouble raising money from investors at all, it could save you to be able to shift toward consulting.

This works better for some startups than others. It wouldn't have been a natural fit for, say, Google, but if your company was making software for building web sites, you could degrade fairly gracefully into consulting by building sites for clients with it.

So long as you were careful not to get sucked permanently into consulting, this could even have advantages. You'd understand your users well if you were using the software for them. Plus as a consulting company you might be able to get big-name users using your software that you wouldn't have gotten as a product company.

At Viaweb we were forced to operate like a consulting company initially, because we were so desperate for users that we'd offer to build merchants' sites for them if they'd sign up. But we never charged for such work, because we didn't want them to start treating us like actual consultants, and calling us every time they wanted something changed on their site. We knew we had to stay a product company, because only that scales.

8. Avoid inexperienced investors.

Though novice investors seem unthreatening they can be the most dangerous sort, because they're so nervous. Especially in proportion to the amount they invest. Raising $20,000 from a first-time angel investor can be as much work as raising $2 million from a VC fund.

Their lawyers are generally inexperienced too. But while the investors can admit they don't know what they're doing, their lawyers can't. One YC startup negotiated terms for a tiny round with an angel, only to receive a 70-page agreement from his lawyer. And since the lawyer could never admit, in front of his client, that he'd screwed up, he instead had to insist on retaining all the draconian terms in it, so the deal fell through.

Of course, someone has to take money from novice investors, or there would never be any experienced ones. But if you do, either (a) drive the process yourself, including supplying the paperwork, or (b) use them only to fill up a larger round led by someone else.

9. Know where you stand.

The most dangerous thing about investors is their indecisiveness. The worst case scenario is the long no, the no that comes after months of meetings. Rejections from investors are like design flaws: inevitable, but much less costly if you discover them early.

So while you're talking to investors, constantly look for signs of where you stand. How likely are they to offer you a term sheet? What do they have to be convinced of first? You shouldn't necessarily always be asking these questions outright—that could get annoying—but you should always be collecting data about them.

Investors tend to resist committing except to the extent you push them to. It's in their interest to collect the maximum amount of information while making the minimum number of decisions. The best way to force them to act is, of course, competing investors. But you can also apply some force by focusing the discussion: by asking what specific questions they need answered to make up their minds, and then answering them. If you get through several obstacles and they keep raising new ones, assume that ultimately they're going to flake.

You have to be disciplined when collecting data about investors' intentions. Otherwise their desire to lead you on will combine with your own desire to be led on to produce completely inaccurate impressions.

Use the data to weight your strategy. You'll probably be talking to several investors. Focus on the ones that are most likely to say yes. The value of a potential investor is a combination of how good it would be if they said yes, and how likely they are to say it. Put the most weight on the second factor. Partly because the most important quality in an investor is simply investing. But also because, as I mentioned, the biggest factor in investors' opinion of you is other

investors' opinion of you. If you're talking to several investors and you manage to get one over the threshold of saying yes, it will make the others much more interested. So you're not sacrificing the lukewarm investors if you focus on the hot ones; convincing the hot investors is the best way to convince the lukewarm ones.

Future

I'm hopeful things won't always be so awkward. I hope that as startups get cheaper and the number of investors increases, raising money will become, if not easy, at least straightforward.

In the meantime, the brokenness of the funding process offers a big opportunity. Most investors have no idea how dangerous they are. They'd be surprised to hear that raising money from them is something that has to be treated as a threat to a company's survival. They just think they need a little more information to make up their minds. They don't get that there are 10 other investors who also want a little more information, and that the process of talking to them all can bring a startup to a standstill for months.

Because investors don't understand the cost of dealing with them, they don't realize how much room there is for a potential competitor to undercut them. I know from my own experience how much faster investors could decide, because we've brought our own time down to 20 minutes (5 minutes of reading an application plus a 10 minute interview plus 5 minutes of discussion). If you were investing more money you'd want to take longer, of course. But if we can decide in 20 minutes, should it take anyone longer than a couple days?

Opportunities like this don't sit unexploited forever, even in an industry as conservative as venture capital. So either existing investors will start to make up their minds faster, or new investors will emerge who do.

In the meantime founders have to treat raising money as a dangerous process. Fortunately, I can fix the biggest danger right here. The biggest danger is surprise. It's that startups will underestimate the difficulty of raising money—that they'll cruise through all the initial steps, but when they turn to raising money they'll find it surprisingly hard, get demoralized, and give up. So I'm telling you in advance: raising money is hard.

Notes

[1] When investors can't make up their minds, they sometimes describe it as if it were a property of the startup. "You're too early for us," they sometimes say. But which of them, if they were taken back in a time machine to the hour Google was founded, wouldn't offer to invest at any valuation the founders chose? An hour old

is not too early if it's the right startup. What "you're too early" really means is "we can't figure out yet whether you'll succeed."

[2] Investors influence one another both directly and indirectly. They influence one another directly through the "buzz" that surrounds a hot startup. But they also influence one another indirectly *through the founders.* When a lot of investors are interested in you, it increases your confidence in a way that makes you much more attractive to investors.

No VC will admit they're influenced by buzz. Some genuinely aren't. But there are few who can say they're not influenced by confidence.

[3] One VC who read this essay wrote:

"We try to avoid companies that got bootstrapped with consulting. It creates very bad behaviors/instincts that are hard to erase from a company's culture."

[4] The optimal way to answer the first question is to say that it would be improper to name names, while simultaneously implying that you're talking to a bunch of other VCs who are all about to give you term sheets. If you're the sort of person who understands how to do that, go ahead. If not, don't even try. Nothing annoys VCs more than clumsy efforts to manipulate them.

[5] The disadvantage of expanding a round on the fly is that the valuation is fixed at the start, so if you get a sudden rush of interest, you may have to decide between turning some investors away and selling more of the company than you meant to. That's a good problem to have, however.

[6] I wouldn't say that intelligence doesn't matter in startups. We're only comparing YC startups, who've already made it over a certain threshold.

[7] But not all are. Though most VCs are suits at heart, the most successful ones tend not to be. Oddly enough, the best VCs tend to be the least VC-like.

Thanks to Trevor Blackwell, David Hornik, Jessica Livingston, Robert Morris, and Fred Wilson for reading drafts of this.

- Russian Translation

THE END