Startup Engineering
Balaji S. Srinivasan
Stanford University

Class: class.coursera.org/startup-001
Signup: coursera.org/course/startup
Forums: bit.ly/startup-eng-forum

Introduction

What is a Startup?

Startups are in. At Yale, freshmen inspired by *The Social Network* are signing up for CS courses in record numbers. At Stanford, a classroom of 75 students signed up 16 million users and made a million dollars in 10 weeks. And at Harvard, a kid left for the Valley and... well, you know that story. Polls show that the vast majority of Americans today love entrepreneurship, small business, and free enterprise even if they disagree about other things. And so the net effect is that Silicon Valley, the geocenter of Startup Land, is the last place in America where people are optimistic. But what exactly are they optimistic about? What is this startup thing anyway?

While different people have different definitions, the general consensus is that a *startup* is a business built to grow extremely rapidly. Rapid growth usually necessitates the use of a technology that invalidates the assumptions of incumbent politicians and businesses. Today startups are strongly associated with Silicon Valley, computer science, venture capital (VC), and the internet. But it was not always so.

Startups in the past

In 1800s America, startups were busy pushing forward the highly nontrivial technologies involved in oil, steel, pharmaceuticals, and telegraphs. In early 1900s America, startups were developing the fledgling automobile, aviation, and telephone industries. Many similarities exist between these earlier startup eras and the present. For example, the Wright Brothers began the multi-trillion dollar aviation industry in their bike store, similar to the canonical garage startup of today. And like early internet companies conventionally used some variant of `.com` in their names, early car companies tended to bear the last names of their founders. Just as Amazon.com survived and Pets.com did not, Ford and Chrysler survive to the present day, while Auburn, Bates, Christie, and Desberon did not survive far past the initial Cambrian explosion of hundreds of startup car companies.

One of the most important features of these early startup industries was the initially relatively low cost of capital required to start a business and the wide open regulatory, technological, and physical frontiers. Take Samuel Kier, the first American to refine crude oil into lamp oil:

> By purifying the oil, Kier figured, he could make a cheaper, easier-to-get illuminant than the whale oil in common use then. He built a simple one-barrel still at Seventh and Grant streets, where the U.S. Steel Tower is now. By 1850, he was distilling petroleum and selling it for $1.50 per gallon. Four years later, Kier enlarged the operation to a five-barrel still that is considered to be the first refinery.

Or Andrew Carnegie:

> Carnegie constructed his first steel mill in the mid-1870s: the profitable Edgar Thomson Steel Works in Braddock, Pennsylvania. The profits made by the Edgar Thomson Steel Works were sufficiently great to permit Mr. Carnegie and a number of his associates to purchase other nearby steel mills. ... Carnegie made major technological innovations in the 1880s, especially the installation of the open hearth furnace system at Homestead in 1886.

Or Eli Lilly:

> Colonel Eli Lilly, a pharmacist and a veteran of the American Civil War, began formulating plans to create a medical wholesale company while working in partnership at a drug store named Binford & Lilly. ... His first innovation was gelatin-coating for pills and capsules. Following on his experience of low-quality medicines used in the Civil War, Lilly committed himself to producing only high-quality prescription drugs, in contrast to the common and often ineffective patent medicines of the day. His products, which worked as advertised and gained a reputation for quality, began to become popular in the city. In his first year of business, sales reached $4,470 and by 1879, they had grown to $48,000. ... In 1881 he formally incorporated the company, naming it Eli Lilly and Company. By the late 1880s he was one of the area's leading businessmen with over one-hundred employees and $200,000 in annual sales.

To say the least, today it is simply not possible for an individual to build a competitive oil refinery on "Seventh and Grant". The capital costs have risen to billions as the industry has matured. And the permitting costs have risen as well; in fact, gaining the necessary permits to build a new refinery - rather than expanding an existing one - is now difficult enough that the last significant new US refinery was constructed in 1976.

And that is a second major important feature: not only were capital costs low, at the beginning of each of these earlier startup eras, there was no permitting process. As one example: in 1921 Banting and Best came up with the idea for insulin supplementation, in 1922 they had it in a patient's arm, and by 1923 they had won the Nobel Prize. A timeframe of this kind is impossible given the constraints of today's regulatory environment, where it takes ten years and four billion dollars to ship a drug. As another example, when aviation was invented, there was no Boeing and no FAA. Thus, from 1903 till the Air Commerce Act of 1926 the skies were wide open without much in the way of rules, with fatal crashes as a common occurrence... and two Wright Brothers could start an airplane company out of a bike store that at one point merged with Glenn L. Martin's company (ancestor of Lockheed-Martin) and ended up as the Curtiss-Wright Corporation, the "largest aircraft manufacturer in the United States at the end of World War II" and still a billion dollar business today.

The initial pioneers in these startup eras thus had time to get underway before competitors and regulations alike started presenting barriers to entry. The messy process of innovation resulted in many deaths from refinery fires, railroad collisions, car explosions, airplane crashes, and drug overdoses. Initially, these injuries were accepted as the price of progress. Over time, however, formal politics arrived in the Wild West, competitors with higher quality arose, and regulations were written that effectively criminalized the provision of beta-quality products. The net result was that the barriers to entry rose sharply, and with them the amount of capital required to challenge the incumbents. Simply from a capital and technology standpoint, it became much more difficult to found a company in your garage to rival Boeing or Roche.

Additionally, collaboration on new rules between large companies and regulators encoded assumptions into law, making certain features mandatory and banning most minimum viable products. The endpoint of industry maturation was that those few potential startups that were not deterred by economic infeasibility were now legally prohibited.

Conceptually, we can think of this process as a sigmoidal curve (Figure 1) replicated in many industries, where an initial Cambrian explosion of startups eventually results in a few pioneers that make it to the top and a combined set of capital/technological/regulatory barriers that discourage the entry of garage-based competitors.

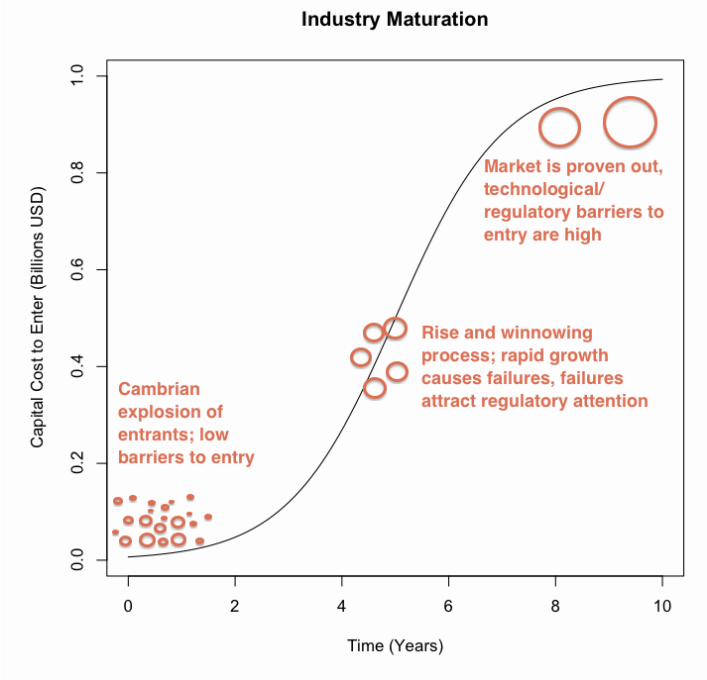

Figure 1: *The sigmoidal curve for the development of an industry.*

Startups in the present

By the second half of the 20th century, the modern era of venture capital and Silicon Valley was well underway with VCs like Arthur Rock and Sequoia, semiconductor companies like Fairchild and Intel, and computer companies like Apple. Networking technology advanced as rapidly as the rest of the computer industry, from the first internet communication to the founding of Cisco, the development of TCP/IP, and finally the invention of the World Wide Web by CERN's Tim Berners-Lee in 1991. At first glance, these industries might seem no different from previous eras, for the costs of directly competing with Intel's fabs, Cisco's installed router base, or Apple's supply chain are surely in the billions of dollars.

But in addition to these mainly *technological* developments, over the 1989-1992 period two things happened from a *market* perspective that may have fundamentally changed the sigmoidal process. The first was the collapse of the Soviet Union and the worldwide turn towards free markets in Eastern Europe, India, China, and the former USSR. The second was the repeal of the National Science Foundation (NSF) Acceptable Use Policy (AUP), which had banned commerce on the internet. The combination of these two events ushered in the

age of a global internet open for business, and thus the *internet startup*: a business which can start in a dorm room and scale to the entire world, accepting payments and providing services to anyone on the planet without need for natural resources, expensive permits, or human clerks.

`</USSR>`: Hundreds of Millions Join the Global Market

From the fall of the Berlin Wall in 1989 to the dissolution of the Soviet Union in late 1991, hundreds of millions of people were suddenly freed to start businesses, practice entrepreneurship, and communicate with their neighbors. To give a sense of the shift, in 1987 photocopiers in the Soviet Union were heavily regulated, with proprietors required to padlock them at night to prevent dissidents from mimeographing material critical of communism. From the Los Angeles Times on Oct. 5, 1989:

> *Soviets Free the Dreaded Photocopier*
>
> In what will be a major step toward implementing President Mikhail S. Gorbachev's policy of free-flowing information under *glasnost*...Inspectors from the Interior Ministry will no longer be checking each machine to ensure that it is locked in secure quarters, that complete registers are kept of the documents copied and that the numbers of copies correspond to the tamper-proof counters required on each machine.
>
> Under the elaborate "instructions on procedures of printing nonclassified documentation," adopted in 1974 to prevent dissidents from photocopying their political tracts, then known as *samizdat*, or "self-published" materials, getting anything copied was never quick and never anonymous. ... "From the first days of Soviet power," the paper added, "even typewriters were put under tight supervision, and that incidentally was abolished only recently."

This sort of total state control over information was characteristic of the entire Eastern Bloc, which is why a pivotal scene in The Lives of Others revolves around the hero's hidden typewriter: contraband in Communist East Germany.

By 1992, though, newly free Estonians from the former Estonian Soviet Socialist Republic could email anyone in the world from their rapidly burgeoning internet cafes, just a few years after the padlocks came off the photocopiers! With the discrediting of communism, other countries quickly followed suit: under Manmohan Singh, India dismantled the License Raj and China redoubled its pursuit of market reforms. The fall of the Iron Curtain thus meant that we entered what has been called the second great age of globalization: a highly integrated global market, where businesses can communicate with and serve customers in other countries as never before.

A simple mathematical model for understanding the significance of these events is in Equation 1. Assume that an exchange or communication requires voluntary participation by two parties. In a theoretically perfect globalized economy, any transaction between two parties is legal. However, if the world were a collection of K North Korea-like states with varying shares of world population proportion p_i, with transactions only permitted within countries $(p_1 \times p_1)$ and banned between countries $(p_1 \times p_2)$, the total proportion of feasible transactions would be:

Figure 2: *North Korea vs. South Korea (from globalsecurity.org). North Korea practices a particular variant of communism called juche, which prescribes complete commercial isolation for the country.*

$$G = \sum_{i=1}^{K} p_i^2 \tag{1}$$

Let's call the $G \in [0, 1]$ the globalization coefficient, representing the proportion of allowed transactions, with $G = 1$ corresponding to a perfectly globalized market and $G \to 0$ as we approach a completely fragmented market of tiny states practicing juche ideology (Figure 2).

At the end of the Cold War, there were roughly 400M in the Eastern Bloc, 1.1B in China, 850M in India, 123M in Japan, 350M in Western Europe, and 250M in the US, with world population around 5.3B. As an extremely crude approximation, assume that the Eastern Bloc, China, India, and (Japan + Western Europe + US + rest of world) could only trade with themselves. Then the proportion of viable transactions before the fall is given as:

$$G = \left(\frac{400}{5300}\right)^2 + \left(\frac{1100}{5300}\right)^2 + \left(\frac{850}{5300}\right)^2 + \left(\frac{5300 - 400 - 1100 - 850}{5300}\right)^2 = .384 \tag{2}$$

While no doubt a very crude estimate, compare this G to the theoretical proportion of 1.0 in a perfectly globalized economy to get some sense of how many new transactions became legal once the USSR, Eastern Europe, China, and India opened for business. Theoretical legality, of course, is different from feasibility - how were these people on different continents supposed to trade with each other?

NSF AUP Repeal: The Internet is Open for Business

During the same period, a debate was brewing at the National Science Foundation. Since the inception of the NSFNET, commerce had been heavily restricted by the Acceptable Use

Policy due to fear of widespread spam, pornography, and malware. In part this was because the culture of the internet was oriented towards academic, government, and military users, whose business models depended on tuition, grants, and tax revenue rather than profits; as such they could see (and could foresee) no immediate benefit from the legalization of internet commerce. With market reforms in the air, however, in 1992 Congress repealed the NSF Acceptable Use Policy, freeing the internet and making it feasible for commercial backbone operators and service providers like AOL to make a profit. Here's Steve Case (founder of AOL) on the atmosphere at the time:

> Next was the pioneering phase (early 80s to early 90s) where a relatively small number of people were trying to build interactive services, but it was an uphill battle. Few people used them (and the Internet itself was still limited to government/educational institutions - indeed, it was illegal to use it for commercial purposes.)

Fascinating! Today, we think of the internet as synonymous with `.com`, with commerce. Yet this was not an inexorable consequence of technological progress, as fairly sophisticated networking in some form had been around for a while. The market was only ready - the global internet market only came into being - once the USSR and NSF AUP left the stage of history.

The Key Features of Internet Startups

Many of the trends that began in the early 90s continue to the present day[1], and are particularly salient features of *internet startups* as distinct from other operations:

- *Operational Scalability*: For the first time in history, it is now feasible for an average person to asynchronously trade with someone across the globe at any time of day without special hardware. When you buy a book from an online store, you don't need to leave your house, interact with a teller, or make the transaction during daylight hours. On the other side of the deal, the store can operate 24/7 without regard for zoning regulations, accept transactions without a McDonald's-size army of store clerks, and offer a storefront on every desktop, laptop, and mobile phone on the planet. The theoretical cost-per-transaction (for both buyer and seller) is thus far lower than any physical store.

- *Market Size*: From an international perspective, more than 2.5 billion people in Eastern Europe, India, China, and the former Soviet Union joined the global market for goods, and subsequently the global internet. But even from a purely national perspective, a business in North Dakota can suddenly market its wares over the internet to the entire country, no longer constrained by mere geography (or protected by it). It's still not trivial to scale across borders or states, but the practically addressable market size has never been larger.

- *Generality*: As we will see, software is the most general product imaginable. Software encompasses ideas (news, blogs), entertainment (music, movies), material goods (3D printing, CAD/CAM), communications (email, social networks, mobile phones),

[1] While the dot-com boom and bust seemed like a big thing at the time, in hindsight it didn't really interrupt any of these trends. People overestimated change in the short run and underestimated it in the long run, something they are so wont to do that Gartner makes a living off of situating technologies on the Hype Cycle.

transportation (electric cars, self-driving cars), energy (smart grids, adaptive braking), medicine (genomics, EHRs), and more besides. This is why Software is Eating the World and all products and professions - from watches, maps, and phones to travel agencies, librarians, and photographers - are being reimagined by software. This is also why it's feasible for Google to go from a search engine into maps or from Amazon to go from a bookstore to a server farm: software engineering skills are highly portable, more so than virtually any other skill save mathematics.

- *Low Capital Barriers*: The hyperdeflation of hardware costs begun by Moore's Law shows no end in sight. The result is that for a few hundred dollars, one can purchase an extremely sophisticated general purpose computer capable of developing arbitrary software and communicating with any other computer in the world. The importance of this can be grasped by analogy to the tool industry. When a mechanic needs a special kind of wrench to fix a car, he needs to order that physical object and hope it's in stock. By contrast, when a computer scientist needs a special kind of tool to fix a file format, he needs only to type furiously into a keyboard to fashion that tool out of pure electrons. In this sense, you own the means of production. You have an incredibly powerful desktop manufacturing device on your desk, a kind of virtual 3D printer before its time, whose price is now within reach of even the world's poorest.

- *Low Regulatory Barriers*: Unlike virtually every physical arena, the internet is still mostly unregulated, especially for smaller businesses. In this sense it is not just a technological frontier, but a kind of physical frontier like the New World, the Wild West, international waters, or outer space: a place with relatively little in the way of formal political control, where code is law. One might argue that this is because of the sheer difficulty of regulating it; given the logic of the globalization coefficient G , a given forum today might pull in assets from servers in three countries and comments from a dozen more, making it difficult to determine jurisdiction. However, the Great Firewall of China, the recent wave of political intervention in internet companies, and the recent SOPA/PIPA and ITU imbroglios indicate the need for continued vigilance on this point.

- *Open Source*: The open source movement dates back to Richard Stallman's launch of the GNU project in the early 80s, but was turbocharged with the advent of the internet and more recently with the rise of Github. The internet is built on open-source technologies such as Linux, Apache, nginx, Postgres, PHP, Python, Ruby, node, Bittorrent, Webkit, and the like. Open source is of particular importance to this course, and it is worth reading this essay on open source economics as distinct from both communism and market economics. Briefly, open source resembles the non-commercial gift economies that some of the more well-meaning proponents of communism envisioned, with the all important proviso that said gifts are not only contributed entirely voluntarily but can be also replicated *ad-infinitum* without any further effort on the part of the gift-giver. As such, they are a uniquely scalable means of charitable giving (e.g. the Open Source Water Pump).

- *The Long Tail*: The sheer scale of the internet enables a second phenomenon: Not only is it possible to address markets of unprecedented size, it is now feasible to address markets of unprecedented *specificity*. With Google Adwords, you can micro-target people searching for arbitrarily arcane keywords. With Facebook Ads, you can reach every single between 27 and 33 with interests in math from small towns in Wisconsin. And

with the right blog you can pull together a community of mathematicians from across the globe to work on massively collaborative mathematics in their free time. Never in the history of mankind have so many (l)earned so much from so few.

- *Failure Tolerance*: When a car crashes or a drug is contaminated, people die and new regulations are inevitably passed. The presumption is often that crashes are due to negligence or profit-driven, corner-cutting malevolence rather than random error or user error. Yet when Google or Facebook crashes, people generally yawn and wait for the site to come back up. This colossal difference in the penalty for failure permeates every layer of decision-making at an internet company (viz. "Move Fast and Break Things") and sharply differentiates the virtual world from the physical world. Moreover, as computer power grows, we can now simulate more and more of the world on the computer via CAD/CAM, virtual wind tunnels, Monte Carlo risk models, and stress tests, "dying" virtually 1000 times before hitting enter to ship the final product.

- *Amenable to Titration*: In addition to all of the above features, it is possible to build a hybrid business in which some aspects remain constrained by the physical world with aspects of the internet startup titrated in to the maximum extent possible. Arguably this trend started very early with Amazon, which combined a virtual frontend with an all-too-physical backend, but it has recently become more popular with companies like AirBnb, Counsyl, Grubhub, Lockitron, Nest, Taskrabbit, Tesla, and Uber as examples to varying degrees. The advantage of engaging with the physical world is that market demand is proven, price points are generally higher, and real-world impact is tangible. The disadvantage, of course, is that some or all of the above trends may no longer work to your benefit. The key is to titrate at the right level.

These trends work together synergistically to produce the phenomenon of the *internet startup*. Huge global market potential means ready availability of large capital investments. Low-cost scalability means growing pains; most processes designed for handling X widgets will have increased failure rates at 3X and complete failure well before 10X. So sustained meteoric growth is allowed by failure tolerance, as the site *will* experience unexpected difficulties as many more customers arrive. Low regulatory barriers mean that good faith technical failures or rare abuses don't result in the business being shut down by regulators, unlike comparable physical industries. Generality combined with low capital barriers means that a company that comes up in one vertical (search or books) can go after an allied software vertical (operating systems or ebooks), thereby delaying the plateau of the sigmoidal curve of Figure 1. And generality allows the most successful companies to titrate into a hybrid vertical (automatic cars or grocery delivery) - causing software-based disruption and making the assumption of stable sigmoidal plateaus in other industries more questionable.

Has the internet fundamentally changed the game? Perhaps the combination of an unregulated (unregulatable?) internet, the low capital barriers of pure software, and the incredible sizes of the global internet market will extend the Cambrian explosion period indefinitely, with software winners moving on to disrupt other markets. That's the bet on continued disruption.

The alternative thesis is that the ongoing financial crisis and possible double dip will affect these trends. Perhaps. One possibility is that politicians begin to exert more centralized control over their citizens in response to fiscal crisis. Another possibility is that the breakup of the EU and a concomitant decline in relative US economic strength will contribute further towards decentralization and a multipolar world. Yet a third possibility is that the recent NSA

scandal makes it harder to scale an internet company, due to national-security-motivated fracturing of the global internet. The key question is whether the internet will change its international character in the years to come, or whether it will prove powerful enough to keep even the mightiest adversaries at bay.

Technological Trends Toward Mobility and Decentralization

All right. We've just talked about the distant past of startups (Cambrian explosions followed by maturity), and the immediate past/present (the rise of a global internet market and the *internet startup*). What does the future look like? Taken as a whole, a number of new technologies on the horizon point in the direction of continued decentralization and individual mobility:

Table 1: *Future (top) and current/ongoing (bottom) technological trends in favor of mobility, decentralization, and individual empowerment.*

Technological Trend	Effect
Realtime machine translation	Reduces importance of native language
Plug computer, Raspberry pi	Reduces importance of size, location
Augmented reality	Layer software over the physical world anywhere
3D printing	Reduces patent/political restrictions on physical goods
MOOCs	Hyperdeflation of educational costs
YCRFS9	Decentralize entertainment
Seasteading	Facilitate emigration, reduce political constraints
Crowdfunding	Reduce importance of Silicon Valley VC, banks
Pharma cliff, generics	Complete disruption of US pharmaceutical industry
Bitcoin	Reduce reliance on single reserve currency, banks
Sequencing	See your own genome without an MD prescription
Telepresence robots, drones	Reduce importance of physical location
360-degree treadmill	Permits actual holodeck; reduces importance of location
P2P/Bittorrent	Reduce reliance on centralized backbone
Blogs	Reduce importance of national opinion journalism
Microblogs	Reduce importance of network news outlets
Social networks	Friends are now independent of physical location
Laptops, smartphones, tablets	Computation is now independent of location
Video chat	Reduce importance of physical location
Search engines, ebooks	Access all information, anytime, for free
E-commerce	Reduce importance of physical store location

The overall trend here is very interesting. Focus simply on social networks for a moment and think about the proportion of your Facebook friends who live outside of walking distance. Since the early 1800s and the advent of the telegraph, the telephone, airplanes, email, and now social networking and mobile phones, this proportion has risen dramatically. One of the results is a sharp reduction in the emotional cost of changing your physical location, as you no longer fall out of touch with the majority of your friends. In a very real sense, social networks impose a new topological metric on the world: for many people who live in apartment complexes, you exchange more bytes of information with your friend across the world (Figure 3) than you do with your next-door neighbor.

Figure 3: *Facebook's Map of the World (from Facebook Engineering). Each node represents a city, and lines represent the number of friends shared between users in each city pair, sampled from about 10 million friend pairs. Lines between cities within the same country are to be expected. But what is interesting are the sheer number of direct lines between individuals in different nations, or faraway cities. Detailed data analysis would likely show more connections between, say, San Francisco, CA and Cambridge, MA than between San Francisco, CA and Victorville, CA - even though the latter is geographically closer and of approximately the same size (100,000 residents) as Cambridge. And today those connections are maintained not on the basis of yearly letters delivered by transatlantic steamships, but second-by-second messages. This is related to the thesis that the next generation of mobile technologies are not location-based apps, but apps that make location unimportant.*

As another example, consider the proportion of your waking hours that you pass in front of a user-programmable screen. For software engineers or knowledge workers more generally (designers, writers, lawyers, physicians) this can easily exceed 50 hours per week of work time and another 10 or more of leisure time. And the primacy of the programmable screen is now extending to traditional industries; for example, Rio Tinto's Mine of the Future project has launched its first autonomous robotic mine, complete with self-driving trucks and laser-guided ore sorters, fully controllable from a remote station packed with computer scientists and process engineers, with on-site personnel limited to maintenance activities and most of the work carried out on a mobile glass screen thousands of miles away. Again the importance of physical location is reduced.

As the importance of physical location and nationality declines, and the influence of software over daily life increases via ubiquitous programmable screens (from laptop to smartphone to Google Glass) we may well be due to revisit the Treaty of Westphalia. Briefly, since 1648, the presumption has been that national governments are sovereign over all citizens within their physical borders. Challenges to the Westphalian order have arisen over time from ethnic diasporas, transnational religions, and international terrorist networks. But the global internet may be the most important challenger yet.

We are already approaching a point where 70% of your friends live outside of walking distance, where work centers around a mobile programmable screen, and where your social life increasingly does as well. Today you have a permanent residence and set a homepage on your browser. The physical world still has primacy, with the virtual world secondary. But tomorrow, will you find your community on one of these mobile programmable screens and then choose your physical location accordingly? This would require mobile screens capable of banking, shopping, communications, travel, entertainment, work, and more... mobile devices which hold blueprints for 3D printers and serve as remote controls for physical locks and

machines. This happens to be exactly where technology is headed.

What is Startup Engineering?

Let's step back from philosophy for a bit and get down to brass tacks. We'll return to philosophy when it comes time to think about business ideas. (Incidentally, this sort of rapid ricochet between abstract theory and down-and-dirty practice is commonplace in startups, like the sport of chessboxing.)

Figure 4: *Startups are like chessboxing. The sport alternates rounds of chess and boxing. Checkmate or TKO to win. (image from Wikipedia).*

For the purposes of this class, *startup engineering* means getting something to work well enough for people to buy. Engineering in this sense is distinct from academic science, which only requires that something work well enough to publish a paper, effectively presuming zero paying customers. It is also distinct from theory of a different kind, namely architecture astronautics, which usually involves planning for an infinite number of users before the very first sale. Between these poles of zero customers and infinite customers lies startup engineering, concerned primarily with shipping a saleable product.

Technologies

One of the primary things a startup engineer does is systems integration: keeping up on new technologies, doing quick evaluations, and then snapping the pieces together. Some people will say that the choice of language doesn't matter, as a good engineer can get anything done with most reasonable technologies. In theory this might be true in the limited sense that Turing-complete languages can perform any action. In practice, the right tool can be like the difference between going to the library and hitting Google.

Below is a reasonably complete list of the technologies we'll be using in the course, with alternatives listed alongside. Note that when building a startup, in the early days you want to first choose the best contemporary technology and then forget about technology to innovate on only *one* thing: your product. Building v1 of your product is unlikely to mean creating (say) a new web framework unless you are a company which sells the framework (or support)

like Meteor. This is why we've chosen some of the most popular and best-tested exemplars in each category: outside of the core technology, you want to be as boring and vanilla[2] as possible until you begin to make a serious profit from your first product.

Table 2: *An overview of technologies used in the course.*

Type	Name	Notes	Alternatives
OS	Ubuntu Linux	Most popular Linux distro.	OS X, Windows, BSD
IAAS	AWS	Used for reproducible development.	Joyent, Nirvanix
PAAS	Heroku	Used for easy deployment.	Joyent, Nodejitsu
Shell	bash	Ubiquitous and reliable.	zsh, tcsh, ksh
Text Editor	emacs	Customizable with strong REPL integration.	vim, Textmate, nano, Visual Studio
DVCS	git	Distributed version control champion.	hg, fossil
DVCS Web UI	github	Dominant among web DVCS frontends.	bitbucket, google code, sourceforge
Language	Javascript	The Next Big Language	Python, Ruby, Scala, Haskell, Go
Macro Language	Coffeescript	Makes JS much more convenient to write.	Iced Coffeescript
SSJS Interpreter	node.js	Async server-side JS implementation	v8, rhino
Backend Web Framework	express	Most popular node.js backend framework	Meteor, Derby
Backend ORM	sequelize	Most polished node.js ORM	node-orm, bookshelf
Database	PostgreSQL	Popular/robust fully open-source relational db	MySQL, MongoDB
Data Format	JSON	The industry standard for simple APIs	XML, protobuf, thrift
Web Server	node.js	node on Heroku is reasonably good as a server	nginx, Apache
Dev Browser	Chrome	Chrome Dev Tools are now the best around	Firebug, Safari Inspector
Scraping	phantom.js	Server-side webkit implementation in JS	webscraping.py, wget
Frontend CSS Framework	Bootstrap	Most popular CSS framework	Zurb Foundation
Frontend JS Framework	Angular	A little simpler than Backbone JS	Backbone, Ember, Knockout
Deploy Target	Mobile HTML5	Ubiquitous, forces simplicity, future-proof	Desktop HTML5, iOS, Android

Over the course of the class, we'll introduce the developer tools first (unix, command line), then discuss the frontend (HTML/CSS/JS, browser) and finally the backend (SSJS, devops). We'll also discuss how to structure your application using a Service Oriented Architecture to make it easy to silo old code and introduce new technologies on new servers.

Design, Marketing, and Sales

To ship a product a startup engineer needs versatility. If you are a founder, you will need to handle things you've likely never thought about. When you walk into an engineering class, the lights are on, the room's rent is paid, and you need only focus on understanding code that someone has hopefully made as simple as possible for instructional purposes. But when you start a company, you're responsible for calling the electrician to get the the lights working, finding the money to pay the rent, and (by the way) pushing the envelope of new technology that no one else understands. At the beginning, you have no product, so you have no money to hire other people to help you with these things. And since you have no product, no money, and the lights are off, it can be challenging to get people to quit their high-paying jobs at Google to work with you for free. So you'll need to be able to produce a passable logo, design the first brochures, and do the initial sales calls - all while moving technology forward.

[2]One point deserves elaboration. If we believe in "boring and vanilla", why have we chosen so many of the newer Javascript technologies rather than using something older like Django, Rails, or even Java? The short answer is that technology stacks evolve rapidly. At the inception of a company, or a new product, you will gain maximum advantage from selecting the best contemporary technology, one that is clearly rising but may have some rough edges, what Steve Jobs would call a "technology in its spring". Today, that technology is arguably a full-stack JS framework based on node.js, making heavy use of AWS and Heroku. In 2-3 years it will likely be something different. But once you pick out the elements of your technology stack and begin developing your product, relatively little added value comes from radical shifts in technology (like changing frameworks), and switching technologies can become an excuse for not working on your product.

Now, if you are the first engineer, or one of the first engineers at a startup that has scaled up, you won't need to handle as many of these operational details. But you will still greatly benefit from the ability to handle other things not normally taught in engineering classes, like the ability to design, market, and sell the product you're building. We will thus cover these topics in the early lectures on market research and wireframing, in the later lectures on sales and marketing, and in asides throughout.

Why Mobile HTML5 for the Final Project?

The focus of the final project is on the use of responsive design to target mobile browsers (like Safari on the iPhone or the Android Browser) for a simple mobile web application, rather than some other platform (like a native Android or iOS app, or the desktop browser). The reason is several fold.

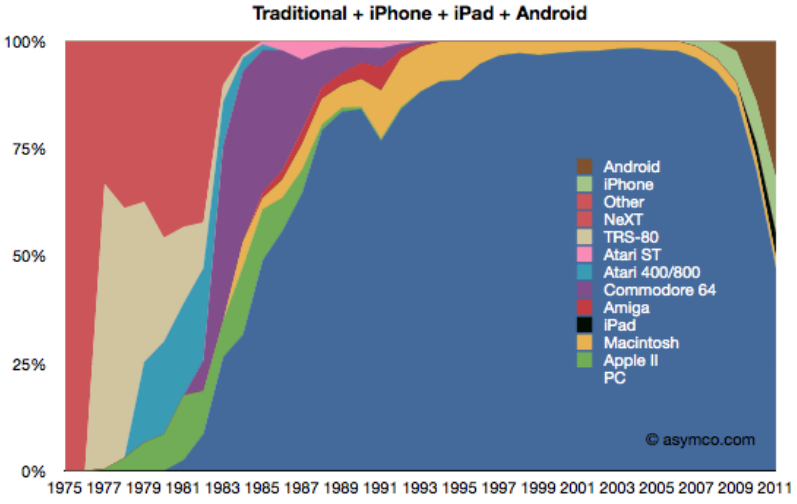

Figure 5: *The Rise and Fall of Personal Computing (From asymco.com)*

- *Mobile is the Future*: First, anyone involved even tangentially with computers should know that mobile clients are now vastly outselling PC clients (Figure 5), a trend that will only accelerate in the years to come, with the proliferation of a host of new clients beyond phones and tablets including wearables, signage, kiosks, Google Glass, plug computers, drones, and appliances.

- *Responsive Design gives Desktop UI for free*: Second, with a modern responsive design CSS framework like Twitter Bootstrap, it's not that hard to simultaneously target a mobile HTML5 client like a smartphone/tablet browser while also building a desktop app. The rule of thumb is that you design for the mobile interface and limited screen real estate first, and then add more bells and whistles (like larger images) for the desktop site. Over time, you might want to actually do a dedicated mobile and desktop site (compare m.airbnb.com to airbnb.com), but responsive design allows you to get an app out the door rapidly and ensure it renders reasonably on most popular clients.

- *Location-Independence increases Scope*: A large class of mobile applications includes many that are location-dependent, like Uber, Foursquare, or Google Maps, which take

your GPS coordinates as a fundamental input. However, as the mobile space evolves, another large class of apps will be those meant to make your current location *unimportant*, e.g. ultra-convenient shopping or banking on the go. Thinking about mobile apps that increase people's location-independence should significantly increase your perceived scope for mobile.

- *Simplicity*: Because mobile screens have greatly reduced real estate, and because users on the go generally don't have the patience for complex input forms, mobile apps force simplicity. This is good for a 4-5 week class project.
- *Ubiquity*: By building an HTML5 app intended for mobile Safari and the Android browser, you will as a byproduct get an app that works in the browser and on most tablet and smartphone platforms. You will also have an API that you can use to build browser, iOS, and Android apps, should you go that route.
- *JS is the Future*: Javascript (aka JS) is truly the Next Big Language. After the trace trees breakthrough in the late 2000s, fast JS interpreters became available in the browser (v8), on the server (node.js), and even in databases (MongoDB). JSON as a readable interchange format is likewise ubiquitous, and "everything that can be written in JS, will be written in JS". Probably the single best reference to get started in JS is Eloquent Javascript.

All right. We now have a sense of what startups and startup engineering is about, and the technologies we'll be using. Time to start coding.