Chapter 1

Intro to Tech Strategy

This is a free chapter of **the Coding Career Handbook**! It is part of the **Strategies** section of the book, and it's yours to keep. Enjoy!

This is a *very* high level overview of tech strategy. That is, the *business of software* rather than the art and science of creating software itself.

It goes without saying that your coding does not have independent value. It must be *applied usefully* on some economic problem to have sustainable real world impact.

1.1 Tech Strategy and Your Career

Perhaps more pertinent to your career: your coding ability will be associated with the success of your company. We infer better ability in someone who was an early engineer at Uber, despite having no knowledge of their actual contributions, compared to someone at an unknown startup who may have accomplished far more interesting things.

Even if you don't care about that, and never intend to be a founder, your understanding of the business you're in allows you to offer suggestions and prioritize work in alignment with economic opportunity. It may not feel like much, but as the person closest to the code, you have a *tremendous* amount of autonomy to the final experience delivered:

- You make delivery estimations, and give advance warning of serious technical blockers
- You make technical tradeoffs between the "quick and dirty" way and the "right" way
- You pick abstractions for scale and pluggability vs rejecting premature abstraction
- You evaluate commercial solutions compared to the cost of a custom solution - the infamous "build vs buy" decision
- You can defensively code for every probable system failure, or understand where failures are more acceptable than their cure
- You sweat the fine details of Accessibility, UI/UX, Uptime and Response time because it matters to users
- You can find the easy-to-implement "low hanging fruit" ideas offered by your data models and pre-existing frameworks (and plugins), and can suggest them to your product owners.

As you advance in autonomy, you will get to pick the projects you work on, and even pitch new initiatives that you eventually own (this is a GREAT career move).

If you read the story of how Google Maps' Satellite view was almost named "Bird Mode" due to a CEO decision, but was ignored by the product team, you will understand how your broad-ranging powers even go as far as *naming* products, which is in no developer's job description. When the chips are down, **Developers are designers and product managers of last resort.**

As you increase in seniority, you will also have to grow in your business judgment. In fact, taking a glance over any **Engineering Career Ladder** (Chapter 26) will tell you how important your business impact is to your career advancement.

Finally - the technologies that you work with are also strongly influenced by their economic incentives. "Free and Open Source" does not mean "Free of any commercial considerations." Nor does it mean "Open Direction decided by Direct Democracy." The platforms you run on , whether it is the browsers, public clouds, databases, payment/fulfilment platforms, or even language distributions, all have massive investments. *Well above 10 figures in some cases!*

1.2 Software is Eating the World

In 2011, Marc Andreesen wrote "Why Software is Eating the World" which set out the foundational thesis of his venture capital firm. **I am assigning this to you as required reading.** It foretold the rise of massive software businesses in the decade since, and made the case for why every industry is now in the business of software. Marc has since updated this with takes on healthcare, biotech and crypto.

This is now widely understood in the software industry, and you should at least be aware of it even if you disagree with it. (Though it does make the software engineer the center of the new world, so you will probably like it a little bit!)

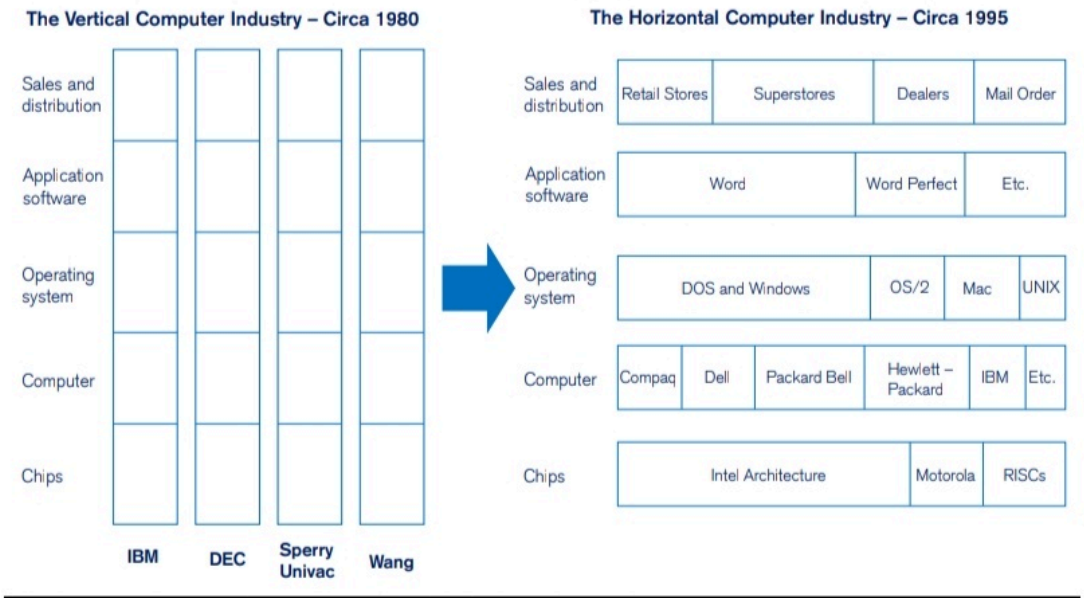
1.3 Horizontal vs Vertical

The basic split in tech strategy is **Horizontal vs Vertical** businesses.

If you work on infrastructure you might be familiar with "Horizontal vs Vertical scaling". This is different. Here we are talking with respect to your customers:

- If your business is designed from the ground up to serve a single industry or customer profile, you are a **Vertical** business. Vertical businesses typically grow by offering more and more capabilities to serve the customers they have, despite these features existing as standalone offerings elsewhere. The goal is to be a "One Stop Shop". User experience can be *mindblowing* when it is vertically integrated - but frustrating when limitations arise.
- If your business can be used by customers across any industry, you are a **Horizontal** business. Horizontal businesses typically grow by reaching more and more kinds of customers, building any integrations required or adding knobs and configs to accommodate their use case. The goal is to "Do One Thing Well" and be the best in the world at it. Many "API Economy" developer tooling platforms are Horizontal: Stripe, Auth0, Okta, Twilio.

The battle between Horizontal and Vertical is timeless in the history of computing. Here is how Andy Grove, former CEO of Intel, described the shift in the computing industry over his career:



New technology categories usually start vertical, because nobody else makes all the components needed. Eventually the individual roles become well-defined enough that horizontal specialists make sense. Clayton Christensen described this as a movement from "interdependent architectures to modular ones". This usually becomes an uncoordinated mess, so then a counterbalancing movement happens and we arrive at an uneasy equilibrium with both horizontal and vertical players in the same industry.

There are two other analogies typically offered to describe this divide:

- **Apple (Vertical) vs Android (Horizontal)**: Apple is 100% vertically integrated for the high end smartphone market. From Apple Pay, iMessage, iCloud, macOS, all the way down to making its own processors. Android is a free open source operating system and it is used by all sorts of phone, watch, car, home, camera, TV, even refrigerator manufacturers.

- **Bundling (Vertical) vs Unbundling (Horizontal)**: a reference to this famous Jim Barksdale/Marc Andreesen quote: "There are two ways to make money in software - bundling and unbundling." Depending on who you ask, we are in the middle or tail end of a Great Unbundling in business software - exchanging the Microsoft Office suite for Airtable, Zoom, Notion, and Slack - before we get tired of jumping between dozens of subscriptions and demand an integrated experience once again.

The maturation of technology is usually rife for a "Cambrian Explosion" of unbundling, which is exactly what happened to Craigslist, Excel, and the chip design industry. Less mature unbundling movements that have only recently begun are Cable TV (splitting up into individual subscriptions like Disney+ and HBO Go), universities (splitting into online courses), radio (splitting into podcasts) and newspapers (though newspapers have already been decimated by social media, top journalists are now leaving and simply starting their own newsletters). There are cases to be made for unbundling everything else from LinkedIn to banks as well. These are all promising areas for startups.

None but the most disciplined of businesses are 100% horizontal or vertical. There is usually a healthy debate within the company as to which direction to pursue, as both are valid ways to grow. But pursuing both can signal a lack of vision and inability to accept tradeoffs and will lead to problems in resourcing, product development and sales/marketing.

You'll also hear this idea applied to other aspects of business - Horizontal vs Vertical Integration, Horizontal vs Vertical Acquisition, and Horizontal vs Vertical Strategy. They are all variants of business expansion along one of these lines.

1.4 Business Models

The next dimension you should be aware of is **business models**. Quite simply, this answers the question of "who pays?", "what directly makes revenue go up?", and (not often enough) "what must the company spend money on?"

The most common ones you should be aware of are *Agencies, Advertising, Subscriptions, and Marketplaces*. Most other companies that employ software engineers have aspects of these embedded within them. I cannot do justice to them in the space I have here, but I will at least introduce them and try to give you enough to learn more.

1.4.1 Agencies

The **Agency** model is the most common one for small teams. I don't have numbers for this, but my guess is it is responsible for most tech jobs as well.

- With an agency model, you have one or more clients and **you are paid for your time**.
- If you are a dev team within a bigger, non-tech company, you are basically an in-house agency with one client.
- If you are a consultant or freelancer, you are a one-person agency.
- There are a thousand small ways to tweak dev setups and payment terms, but broadly, as the amount of dev-hours grows, so does the

amount of money flowing into the agency.

Ideally, you should be trying to get the most done per hour, but cynically, bad incentive systems can lead to just booking more hours. The common thread is that your income isn't fully pinned to the success of your client's business, which is sometimes a feature and often a bug of the Principal-Agent Problem. Despite the flaws, agencies are still so popular because of the sheer amount of work that needs to be done, and the specialized talent needed for certain types of high skill work.

1.4.2 Advertising

The **Advertising** (or Media) model is next most common. Here you make money by increasing the traffic to your site or usage of your product and then selling advertiser spots.

- Sometimes the advertising is *display ads* (paying for Cost Per Milles - aka ears and eyeballs - just to be present and build brand awareness).
- But most advertisers these days prefer *performance based* marketing - paying for directly measurable user actions e.g. Cost per Click.

Most social networks and news/opinion sites run this way, though there is an absolutely massive assortment of marketing technology to help ad buyers find the best ad inventory. Because end users pay nothing and advertisers pay for access, the derisive view is that "Users are the Product." However this may not always be a negative - the Wirecutter and the Points Guy are both well-regarded high quality content sites

that make their money from affiliate marketing, which is a variant of performance based marketing (pay-on-purchase rather than pay-per-click).

Social Media and Digital Media differ in one important way - digital media companies have to hire journalists and editors to create the content that draws people, whereas social media companies get **User-Generated Content** for free.

Social media is a unique intersection of tech and society, where we trade our information and attention for news and social status. On paper you might be an ads business, but as far as humans are concerned, you offer Status as a Service. **People literally compete to give you their best content for free**. This sounds like a wildly attractive proposition, and at first it was. You can see Reid Hoffman's Series B pitch deck for LinkedIn to appreciate just how compelling it is. But in recent years the hidden "cost" of running a social media company has emerged in the need for content moderation. You might start out a technologist and end up spending all your time debating the fine points of 47 U.S.C. § 230 and having complex debates about deplatforming and misinformation.

With digital media companies, including podcasts, the trend has been toward realizing that ads aren't the only way to make money - you can charge your audience directly! That leads us to subscriptions.

> Tip: The reality of most media companies today is to tend toward some sort of hybrid Advertising (for free users) and Subscription model (for highly engaged users), so it isn't an either-or proposition.

1.4.3 Subscription

The next most common type of business model is **Subscriptions**:

- If you sell usage of your software, this is known as Software as a Service (SaaS), which is an investment category all its own. The common characteristic of the IaaS/PaaS/SaaS models is they transform Fixed Cost to Variable Cost which provides immediate value for customers.
- Content subscriptions are the other major category. This includes audio (e.g. Spotify), video (e.g. Netflix), news (e.g. the New York Times), blogging (e.g. Stratechery), data (e.g. Crunchbase) or membership to a professional group/community. All of which require software to support them.

Since users pay directly for the software/content/membership, and can walk away at any time, the incentive alignment is clear - use subscription revenue to make a better offering, which helps drive more subscriptions, which helps finance a better offering, and so on. Since digital content can be replicated infinitely, the gross margin and therefore cash flow of these kinds of business is high. To grow, the business has to expand its marketing funnel, increase conversion rates, keep a lid on cost of content (e.g. revenue sharing with content creators), and decrease churn.

Most subscription businesses are a buffet - pay your subscription and it's all-you-can-eat. This has an inherent flaw - some people just eat a whole lot more than most. This is expensive to support and the lighter users subsidize those who "abuse" the platform (by bringing down average usage). Therefore all subscription business eventually start charg-

ing per-seat, and then find their way toward some form of metered billing (using some form of value metrics). Doing this too eagerly can have the perverse effect of punishing your most engaged users.

> Fun fact: Marc Benioff is widely credited with inventing SaaS, and ironically introduced it with the famous "No Software" campaign in 1999.

1.4.4 Marketplaces

Marketplaces are the hardest software businesses to build, and therefore there are fewer of them than other kinds of business. However, once established, they exhibit gobsmacking double-sided network effects, which makes them very valuable (Bill Gurley describes marketplaces as creating "money out of nowhere"). Developers both use and work for marketplaces every day without realizing it - ranging from Airbnb and Uber to Cameo and Udemy (see this list for more).

Marketplaces match buyer and seller, just like their offline counterparts. The marketplace gives both sides an assurance of:

- **liquidity** (buyers will be able to find what they want, sellers will be able to sell what they have, and both can do it faster than anywhere else)
- and **quality** (buyers are good customers whose checks don't bounce, and sellers must not sell fake or defective products, or they get kicked off the platform).

In exchange, it takes a fee from either the buyer or seller or both. Because the fee typically is a percentage of the money that changes hands, this is called a take rate and marketplaces want to grow the Gross Merchandise Volume it is based on. Take rates range wildly based on platform power - Gumroad charges 3.5% while Apple and Google's app stores take 30%.

This model sounds simple, but there are a lot of ways to make additional revenue. For example, suppliers often pay certification or listing fees, or they could instead *be paid* to join. It also turns out that a marketplace's own site is prime ad space and that customers will pay for better service. So as your marketplace grows up the Hierarchy of Marketplaces, you can build both an entire advertising business AND an entire subscription business INSIDE a marketplace business, which is what Amazon has done.

In a way, this is the business model to end all business models, because you essentially now run your own economy.

Most platforms eventually add marketplaces, sometimes called App Stores. Fun fact: Steve Jobs gave Marc Benioff the idea for Salesforce AppExchange, the first "App Store" in 2005.

Two final, major advantages you need to know:

- Marketplaces don't own inventory, since suppliers are the ones to bring their inventory to market. This makes them asset light, which means they can scale enormously with little investment. Airbnb offers more room nights than any hotel chain in the world, without owning any hotels. Uber and Lyft transport more passengers than any taxi company, without owning a car.

- Large enough marketplaces drive both seller and buyer to **optimize for each other** - buyers want high ratings (especially when buying repeat *services*) and sellers want great reviews. The slightest nuances of the platform - everything from picture dimensions to product offered - will be exploited to exactly meet the marketplace's needs. This not only means that buyers and sellers are optimizing for each other for free, it also makes starting a competitor marketplace extremely difficult since investments have been made and reputations gained. In other words, great marketplaces have positive virtuous cycles.

These benefits aren't free - **marketplaces are hard to build** for a few reasons:

- **Fakes and Disputes**: Marketplaces offer an implicit or explicit guarantee of quality, which means they need a way to handle what happens when things go wrong.

  - If goods are sold, you must handle the issue of fakes, lemons, and returns. This might not seem fun, but Zappos made great return policy a competitive advantage.

  - If services are sold, you must handle the billion things that can go wrong when humans work for humans, and you're not around to verify what actually happened. Airbnb had to roll out a Million Dollar Liability Insurance Program to reassure hosts, whereas Uber uses its rating system to keep drivers in line.

    The art of maintaining marketplace quality is an entire discipline of its own. Underlying the entire discussion is the fundamental problem of how to **create trust between strangers**. The best place to learn more is Lenny Rachitsky's research.

- **Cutting out the Middleman**: Every strong supplier eventually chafes at paying the take rate. At stake is not only more revenue, but also a more direct, long term relationship with the customer, free of any unfavorable changes the marketplace may make in future. For service marketplaces, buyers and sellers who like each other enough can simply take their relationship "offline". This means that buyer and seller *churn* (also known as *platform leakage*) is a huge problem for marketplaces, and it must provide a compelling reason for both sides to stay on even after they have found each other. Two natural reasons to stay are polar opposites: either infrequent, large transactions where reputation matters (Airbnb) or frequent, small ones where you don't care who does it in a form of perfect competition (Uber).

- **The Chicken-and-Egg Issue** (alternatively, the "cold start" problem): If there aren't enough buyers, it is not compelling for suppliers to join the platform. If there aren't enough suppliers, buyers won't even come by. A marketplace can toggle back and forth between being demand or supply constrained a few times in its life. As many as 40% of marketplaces remain supply constrained throughout their entire life, which is why having an encyclopedia of ways to grow marketplace supply is a hot topic.

- **Political Risk**: Successful marketplaces are very disruptive to the livelihoods of many legacy sellers because they commoditize their offering. There is almost always political backlash because of the tremendous power shift. If lobbying is successful, your marketplace could be destroyed by legislative fiat.

One way to get past many of these issues is to be your own supplier - so that you only grow the demand side of the market. This is sometimes called "single player mode". If your "marketplace" has value without

any network effects, then it has a reasonably good chance of attracting enough people to *get* network effects (Often referred to as "come for the tool, stay for the network"). You could view all ecommerce businesses as "one-sided marketplaces", although the trendy term for this is Direct to Consumer or "D2C".

1.4.5 Gaming

In reviewing this section, Julian Garamendy pointed out one other business model that wasn't quite accounted for: Gaming. Given that gaming is a huge megatrend (we discuss **Megatrends** in more detail in Chapter 29), I feel I must discuss this somewhat. Some games, like World of Warcraft, are subscription businesses. Others, console and PC games like Zelda and Red Dead Redemption, are one-off purchases, akin to D2C ecommerce with entirely virtual goods.

But there is a massive genre of games which consist of microtransactions and real money auction houses - games like Fortnite, Candy Crush, Star Wars Battlefront and Diablo 3. These rely on "whales" getting hooked, and are more akin to casinos. They may not be the most beneficial software humanity makes, but they generate enough money that they must be acknowledged.

1.5 Platforms and Aggregators

I've used the word "**Platform**" a couple times without definition. The word is horrendously overloaded, so that you can never be sure what

is meant without more context. When it comes to the business of software, though, you can do a lot worse than listen to Chamath Palihapitiya quoting Bill Gates:

> I was in charge of Facebook Platform. We trumpeted it out like it was some hot shit big deal. And I remember when we raised money from Bill Gates, 3 or 4 months after — like our funding history was $5M, $83M, $500M, and then $15B. When that $15B happened a few months after Facebook Platform(...) Gates said something along the lines of, "That's a crock of shit. This isn't a platform. **A platform is when the economic value of everybody that uses it, exceeds the value of the company that creates it. Then it's a platform.**

He's right. The Bill Gates Line defines Platforms as serving an ecosystem that exists because of them, but is also much bigger than them. And because Platforms are such tremendous economic centers of gravity, we need to differentiate them from your average, run-of-the-mill marketplaces (which, I hope I have established, are powerful economic engines already).

The three most important platforms of all time are Windows, iOS and Android. Per Ben Thompson, it's no coincidence that all are operating systems:

- **Windows was the OS for the Desktop Age**: Developers made apps to run on Windows and manufacturers made hardware to run Windows, which made Windows more attractive for both sides. Quoting Ben:

> The end result was one of the most perfect business models ever: commoditized hardware vendors competed to make Windows computers faster and cheaper, while software developers simultaneously made those same Windows computers more capable and harder to leave.

- **iOS and Android are the OS for the Mobile Age**: Developers make apps for iOS and Android, and users are trained to find everything via the Google and Apple Play Stores, which makes iOS and Android more attractive for both sides.

> Note: Ben originally called Google Search a platform, but changed his mind as he developed his theory further.

Because of its unprecedented success as the main cross-platform platform for gaming, Epic Games' Unreal Engine must also be considered in the same league. As with all things Gaming-related, see Matthew Ball's *epic* Epic Games primer for more details.

Platforms exist on a higher level than business models - for Windows it was licensing, while for Google it is ads. Yet the two-sided nature of a Platform makes it seem like a Marketplace, and Bill Gates' definition seems comparable to GMV.

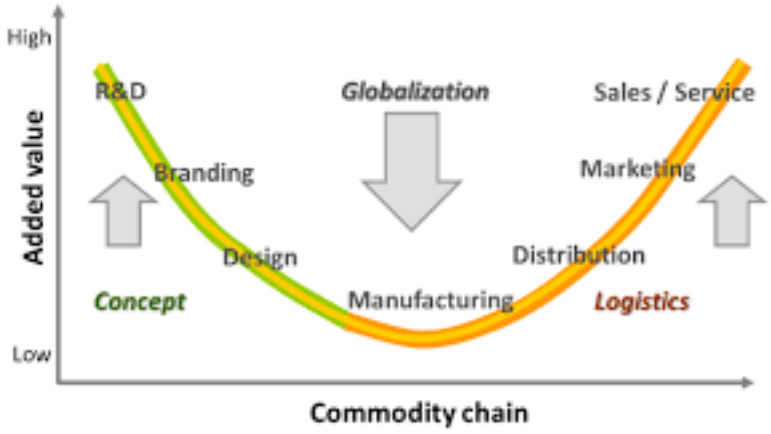
But Platforms don't play the transaction volume game - their M.O. is to look at the most critical use cases and to build them out as subsequent products. Windows built out Office (Word, Excel, Outlook, etc), and then Windows Server. Google built GSuite (Docs, Sheets, Gmail, etc), and acquired YouTube.

I mentioned that the usage of the term "Platform" is quite overloaded. Hugh Durkin identifies three traits:

- **serving multiple different types of consumers**
- **facilitating efficient value exchange through building and operating a marketplace**
- **building a shared set of common standards through standardised internal and external facing APIs**

Search around and you'll find more definitions. Everyone tries to put their own spin on it, which mainly serves to show that it's an important thing to be.

Not all Platforms are economic powerhouses. If the market is not a natural monopoly, then there may be a lot of competitors, and a race to the bottom in terms of pricing. This happens in many industries from publishing to semiconductors. In fact there is often a phenomenon where platforms become the *least* value-adding part of the supply chain, with relative power flowing to creators and distributors. Stan Shih, founder of Acer, famously called this the Smiling Curve, and the concept has been extended to everything from self-driving cars to Taylor Swift (aka the music industry).



For platforms to do well, they have to constantly add value or achieve unbeatable network effect. Some financiers have seen great success reducing competition by rolling up a group of platforms.

A contrasting economic model to Platforms are known as **Aggregators**.

1.5.1 Aggregators

Aggregators are the main characters of Aggregation Theory, defined by Ben Thompson.

> Note: If this seems strangely focused on the theories of one person, it's because Ben has shaped the entire zeitgeist of tech with this theory, to the point of being quoted at Apple keynotes - so I feel *compelled* to introduce it to you.

Aggregators must have three characteristics:

- Direct relationship with users (payments, accounts, regular usage)
- Zero Marginal Costs for serving users
- **Demand-driven** Multi-sided Networks with *Decreasing* Acquisition Costs (aka a very specific type of the two-sided network effect we have discussed)

Aggregators take advantage of a fundamental shift in power enabled by the Internet and the digital economy. Because the marginal cost of digital goods is zero, the ability to generate profits has shifted from companies that control the distribution of scarce resources (Suppliers) to those that control **demand** for abundant ones (Aggregators).

If you *aggregate users*, you call the shots. This means great user experience is paramount, and explains the tremendous amount of investment in web and mobile clients over the past ten years. (One answer to why React developers are in such demand.)

Levels of Aggregators:

1. **Supply Acquisition** - they have a great user relationship, but buy their supply, e.g. **Netflix** and **Spotify**. Content cost is a concern.
2. **Supply Transaction Costs** - they don't buy their supply, but pay some marginal costs to bring suppliers onboard, e.g. **Uber** and **Airbnb**.
3. **Zero Supply Costs** - they don't buy their supply, and incur no supplier acquisition cost, e.g. **Amazon**.
4. **Super-Aggregators** - they have at least *three* sides - users, suppliers, advertisers, and have zero marginal costs on all of them. E.g.

Facebook (with Instagram), **Snapchat** and **Google**.

1.5.2 Platforms vs Aggregators

It can help to situate these two models by contrasting them. I'll point you to Ben's writeup on his site:

Platforms (e.g. Windows) are critical for their suppliers (e.g. Windows apps) to function, Aggregators (e.g. Google) aren't critical for their suppliers (e.g. websites) to function.

Platforms facilitate a relationship between users and 3rd-party developers, while Aggregators intermediate the relationship between users and 3rd-party developers.

Platforms help people do things (aka Bicycles for the mind), Aggregators do things for people.

To find more information, you can read Ben Thompson's body of work - be aware, his definitions changed between 2015 to 2019. Also, the commonly accepted usage of the word "Platforms" also encompasses Aggregators.

One final point is relevant for us - Both Platforms and Aggregators make it so much easier for suppliers to reach customers that it enables new types of businesses to be created atop them. Apple, Microsoft, YouTube, Amazon, Teachable and others have all minted millionaires, and developers can make a great living working on them or for them.

1.6 Other Strategic Perspectives

As you might see, the analysis of what drives the rise and fall of tech companies can get very nuanced indeed. Though tech giants get all the limelight, good ideas are fractal, and you can apply them in smaller contexts within your professional network, language ecosystem, and internal company politics.

I haven't any room left but want to share a few more interesting dynamics you can investigate on your own:

- **The funding of Open Source** is always a point of contention - Open Core models are increasingly viable and benefit the developer ecosystem while also being great employers. However, if your core is open, anyone can compete with you on hosting your core, so this has caused the Great Relicensing.

- **Land Grab vs Organic Growth**: some business opportunities must grow rapidly due to a Winner-Takes-Most network effect, and so should raise VC. Others will always be one of many so profit and unit economics should be a core focus for bootstrapped growth. Joel Spolsky has a good introduction to this idea.

- **Categorical Imperatives**: I have a suspicion that software has intrinsic desires, expressed by inevitable and unimaginative customer and product manager feature requests. If you anthropomorphize the codebase you are working on and treat it as a living, breathing thing, you can think about what IT "wants". You can therefore predict what features you are going to have to build. Examples:

- Every collaboration app wants email
- Every data analysis app wants the power of Excel
- Every marketplace wants to be two-sided
- Every social app wants chat
- Every User Generated Content app wants Snapchat's Stories format
- Every B2B app wants a dashboard
- Every site author eventually wants a CMS