How startups beat incumbents

by Jason Cohen on February 25, 2024

A startup can beat a large, successful incumbent, if it does things the incumbent can not or will not do. Here are those things.

It doesn't seem possible for a startup to beat an incumbent.

An incumbent has everything: money, brand, customers, a sales team, marketing that generates thousands of leads every month, product and engineering teams that constantly ship. They mine their big existing customer base for ideas, and then build exactly the right features, and then charge for it. Their 24/7 support team provides faster and better service than someone working in their pajamas at home. They don't have to build the basics or ask Twitter how to manage international sales tax. They can just focus on innovating.

Of course if you've ever worked at a big company, you know that **while those things seem true, it often doesn't feel like it.** Big companies are rarely well-oiled innovation machines, and it certainly doesn't feel like you're constantly outpacing the competition.

When we analyze how incumbents are vulnerable, we uncover opportunities that startups can exploit to win, where there's often nothing the incumbent can do about it, despite their advantages:

- Taking risks that cannot be quantified
- Addressing a profitable niche
- Doing delightful, valuable things that don't scale
- Unsurpassed customer service
- Leveraging new technology
- Having an opinionated personality
- Doing things that aren't zero-sum
- Being worse-but-acceptable in most dimensions
- Being low-cost against a profit center

The pattern: Every big-company advantage creates exploitable weakness

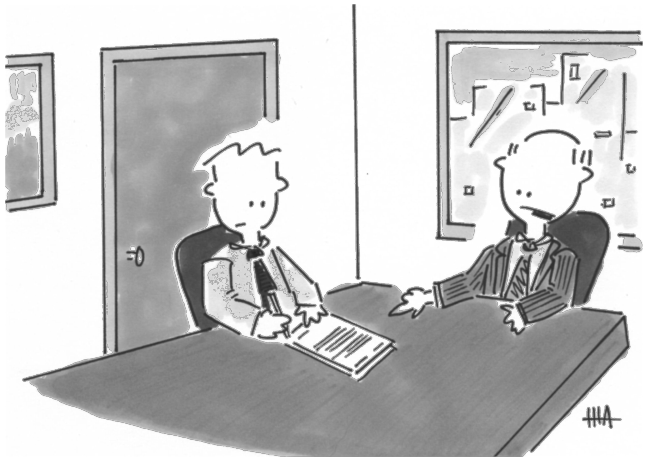
The reason big companies don't function as well as described above is that things at scale are super-linearly more difficult.

It's an advantage to have 100,000 customers when you're figuring out what the next feature should be, or when you're launching a second product, or when you get free growth from word-of-mouth.

But it's a disadvantage to have a lot of customers when you want to innovate with your product, because no customer wakes up in the morning and says: *Gee, I hope the software I'm accustomed to dramatically changes today*. Customers don't want to learn new UIs. Customers have workflows that you have to accommodate. Old technology that powers those 100,000 customers doesn't support the latest technology. You have to update documentation and videos and the people in support and sales who need to be retrained. Even a simple change can be difficult and expensive, and certainly low-ROI.

Besides "scale," a big company must accommodate things startups can ignore.

There's the legal department, for example. A startup does all kinds of illegal things. Most startups do not pay taxes properly, sometimes not at all, especially in other countries. Startups don't adhere to all the Acceptable Use Policies of all the products they use. Startups don't have a security team who vets vendors before sending them sensitive data, or vets libraries before they're integrated into the code base, causing all of their supposed "secret intellectual property" to become open-source.



"This just says you won't reveal anything about our nondisclosure agreement."

As a result, the startup not only moves more quickly—which is how most people characterize it—but they can completely skip things that a larger company cannot. So Uber decided to just do illegal things in order to grow. An incumbent taxi company obeys the law, so they lose. You could say that that's not fair. You could say that's what regulation ought to prevent. But the reality is that startups often ignore the law, and that can be an edge.

The way a startup wins, is to do things that incumbents cannot or will not do.

So, let's see how to attack where they cannot defend.

Take risks that cannot be quantified

The way a larger company decides to take a risk, such as launching a new product line or entering a new market, is by creating a detailed analysis of the opportunity, and a cost estimate. Then the decision is:

1. Is this is a good ROI? *(potential-revenue divided by costs)*
2. Do we have conviction that the risk of failure is low?

How can a startup exploit this decision process?

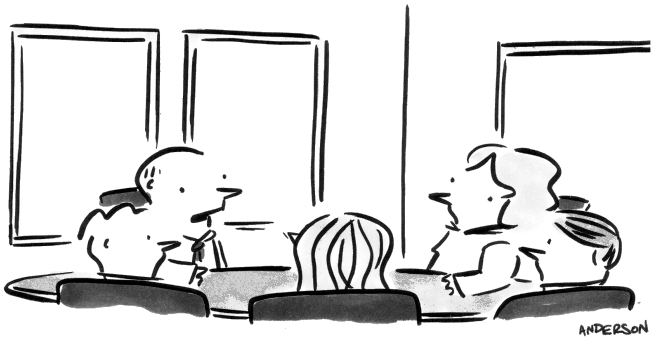
Starting with decision (1), the analysis is typically wrong. There are studies everywhere—and your own experience, if you've worked at a large company—showing that most development projects are significantly late and over-budget, and also that the outcome is typically worse than expected. Both sides of the ROI fraction are worse[1].

[1] I cover exactly what to do about this in an article about how to do ROI analyses correctly.

So, whichever projects *appear through traditional cost-benefit analysis* to be low-ROI, are unlikely for an incumbent to do, even though there's a good chance that (a) they're rejecting genuinely good ideas and (b) they're accepting weaker, more straightforward ideas, only because those more readily lend themselves to ROI analysis. A startup who selects "apparently" low-ROI projects, will probably have no competition from incumbents.

With decision (2), big companies don't like to take big risks *even when the outcome might be large*. The fear of failure is an order of magnitude more motivating than the desire to innovate or even the greed of success. One reason is that the core business is probably going well, and you "don't want to mess that up." Another reason is that no one wants to be the one who proposed, fought for, and then presided over a multi-million-dollar failure. Another reason is that none of us really knows what a "big risk" is anyway, because even experts can't predict what will happen, and we don't know how to talk about risks or measure risks, even in retrospect.

At a big company, it's safer to say "let's gather more data" and "let's wait for consensus" than it is to take a risk.



"OK, now that we all agree, let's all go back to our desks and discuss why this won't work."

But innovative things are often high risk or unknown risk[2]. Therefore, a startup can pick things which are risky, or where the risk is impossible to ascertain, but where the potential upside is high, and incumbents are unlikely to follow.

[2] "High risk" means we know number that is the probability of success, and it is low. "Uncertainty" is unknown risk, meaning we don't know what the number is at all. Though different, they have the same result in this context: It's too scary to take the chance.

Indeed, this is also what I recommend for work-planning in general in selecting and prioritizing Rocks versus Pebbles, where the "big Rocks" that move the needle should be selected on the basis of potential outcome rather than ROI, whereas the Pebbles should indeed be based on ROI.

Startups can do it, while incumbents are almost always too fearful.

That's OK, the incumbent can buy you later at 10x revenue.

Address a niche

An incumbent wanting to expand into a new market or launch a new product line, must apply a significant amount of money and people—a large investment, even for them. What kind of return do they need on that investment?

The answer is: It has to materially affect their growth rate.

The rule of thumb is it should increase their overall revenue by at least 10%; some people call this "the materiality threshold." However, that number goes up as the (perceived) risk or (actual) investment increases. For an incumbent with hundreds of millions in revenue, that means the product line must have a good chance of making $50-$100 million, or they won't even try (nor should they).

> *"Given our size, we only see a few good things [to invest in]. If we were smaller, then we'd see lots of good things."*
>
> —*Warren Buffett.*

Another financial metric that creates a materiality threshold is a certain ratio on the Profit & Loss statement.

A software company at scale spends around 20% of their revenue on R&D, which includes Product, Engineering and Design. A fully loaded team, including salaries, taxes, software, hardware, training, management, travel, office space, and so on, can cost upwards of $2 million a year[3]. If that cost is supposed to be 20% of revenue, the team needs to generate at least $10 million in annual revenue, which means the product they're working on must do that, even if the product is a small add-on or the team owns a subset of a larger product.

[3] Obviously this varies by geography, employment laws, tax laws, and the size and composition of the team; this estimate assumes a team of eight engineers, a product manager, a designer, and a manager, in the US, and rounding-off for rhetorical simplicity.

Of course, a small startup doesn't see it that way. A team of two founders and one employee isn't thinking "We have to make $10 million a year, otherwise it's a failure!" This means the startup can focus on a niche that doesn't need to generate $10 million; it could generate $1 million.

The startup can focus on a niche and ICP that a big company cannot afford to target, either because that niche wouldn't hit the overall revenue materiality threshold, or wouldn't hit the one-team P&L threshold.

With that focus, the startup has no direct competition from an incumbent. In fact, the larger the incumbent, the less the startup has to worry about competition.

Feels good.

Do things that don't scale

WP Engine was the first in our market to support LetsEncrypt[4]. We knew nearly all of our customers would want it. We wanted to promote it heavily, but we were already deep into scale, with 100,000 customers who could potentially use it on day one.

[4] If you're unfamiliar with this web technology, don't worry—the details aren't important to the story. Suffice to say: It was a desirable capability, and is now ubiquitous.

Therefore, it had to be scalable before we released it. That means breaking down the components into queues, in case one step in the process was faster than another, or in case one failed and had to be repaired before the system could make progress. And we had to monitor those queues, and send alerts to humans if it stayed broken for too long. And we had to run at-scale tests to make sure it worked when there were 1000 requests simultaneously with random failures. And we had to train hundreds of folks in tech support on the questions we anticipated, and train hundreds of folks in sales on how to leverage this to make sales, and work with marketing on how to message it. And we had to make sure it had close to zero bugs, because if thousands of people start using it, and 10% ran into a bug, we'd crush our support team, and hundreds of people would take to Twitter to complain.

We were correct to invest many months of time in all these areas; on the first day, thousands of people *did* start using it, tens of thousands in the first month, and indeed some components did break, and lots of people asked questions in tech support. And people praised us on Twitter as a result.

The good news about a large customer base, is that you can have 1000 users on day one and 10,000 on day 30. In this case we gave it away for free, but in general an incumbent can be generating $1M ARR or even $10M in short order. A startup cannot do that.

But a startup can launch something in a few weeks and iterate. We had to be heads-down for most of a year. And so another startup opportunity emerges: A startup doesn't have to operate at scale.

You might say: A larger company can still make an alpha version of a product, show it to a few dozen customers, and iterate from there. Indeed, we also do that; it's a wise process. However, what a larger company is *not* willing to do is take the next couple of years just to get to $1M ARR and then take a couple more years to get to

$5M and then take a couple more years to get to $20M. That's just way too long to get the "material" amount of revenue and does not leverage their at-scale assets.

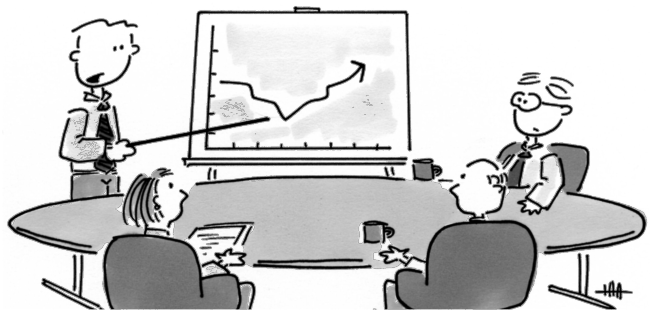
But the ramp I just mentioned is ideal for a startup; in fact, that is a highly successful growth rate. It allows time for the product to settle in and slowly get to the point where it is scaling. This is a reasonable, fun, and plausible path. No one knows you exist; that's bad for sales, but good for iterating without harming your reputation.

Therefore, if a certain product idea is naturally very easy to scale from day one, that's easy for an incumbent to copy. But if the product is naturally difficult to scale from day one, that's ideal for a startup.

Unsurpassed customer service from founders & engineers

I personally handled support tickets every day at WP Engine until we had around 35 people; this is typical for a customer-oriented founder. There's no better way to understand how people interact with the product than to talk to them about when it's going wrong or not meeting their expectations. You get feature ideas, you understand how to streamline the product and how to increase retention.

Customers are impressed by the quality of support and the range of problems you can solve. They won't get that from a company with a thousand tech support reps. Some startups aren't interested in providing great support, but those that do are naturally and even effortlessly orders of magnitude better than a large incumbent. It's a competitive advantage available to everyone.



"So, as you can see, customer satisfaction is up considerably since phasing out the complaint forms."

As if those benefits to both customers and product development aren't enough, it also fosters real love and loyalty from customers. That love translates to forgiveness when you do have problems; see the mountain of supportive tweets when a small-but-lovable company has a big outage or security issue. It also translates into word-of-mouth advocacy, as customers naturally reciprocate, and thus great support results in inexpensive growth. Love is the best form of "willingness to pay."

Which incumbents cannot compete with.

As a startup scales, it loses this advantage. I distinctly remember each time in the past 14 years at WP Engine that a new competitor would brag about their amazing tech support. While WP Engine continues to objectively[5] have world-class support, it's not the same as personal attention from the founder of the company!

[5] Won a dozen Stevie Awards, and maintains 98% CSAT.

You could decide to never to grow that large, optimizing for profit and efficiency rather than revenue and scale, and make Support a permanent competitive advantage.

No matter which future you pick, this is still a great way to get started.

Leverage new technology

When a technology is new, the risk of using it is high.

Maybe it won't be supported in five years. Maybe you won't be able to hire dozens of engineers who are familiar with it. Maybe[6] it will have a big security problem. Maybe it works well for the "Hello, World!" case, but doesn't work at scale. Maybe it's efficient for one developer but too difficult to coordinate with thirty. These are all reasons why incumbents are absolutely correct in avoiding new technology.

[6] In fact, *certainly* it will have *many* security issues… the question is whether they will be identified and addressed.

But a startup doesn't have these concerns. Not because the big company is wrong, but because the constraints are different. The thing that will kill the startup is not going to be the tech stack; it's going to be that it's too hard to find customers, or they don't have a budget for this problem, or it's too hard to compete, or you run out of money, or any of the other things needed to get to Product/Market Fit.

And new technology often creates a competitive advantage. New technology makes certain things efficient, or enables things which previously were impossible, as in the current case of AI.

The startup *is* taking a risk on that technology. If you're banking on a new open source project, and it doesn't take off, you might have a product built on a platform that is no longer supported, and that's bad even if you're a small startup.

However, what is the worst case, assuming the startup isn't already dead by then? It's that, five years from now, you've built a sustainable company, and now you have to redo your platform using different technology. That does really suck for you. You might have to pause new features for a year to make the transition, and engineers, product managers, and sales reps alike will hate that. But if this penalty "buys" you a successful company, then it was worth it.

Have an opinionated personality

It is rare for a large company to express a personality.

There are many reasons for this. They want to address a large and therefore diverse market to sustain their revenue and growth; by speaking to everyone, they speak to no one in particular. They have an established brand, which creates trust, which is one of the reasons they win sales, embodied in the now-outdated phrase "No one ever got fired for hiring IBM." So the language on the homepage and inside the product needs to reinforce this trust, which means being impersonal but solid.

Another reason is that customer communication is spread over hundreds or thousands of people, from Support to Sales to Marketing to Product to Design. There isn't such a thing as a genuine, human personality and style that a thousand people share and express identically. Whereas everyone can conform to generic but professional language.

But a startup doesn't have these constraints. Indeed, the founder often has a strong personality, with specific ideas of what's good and bad and how things should be done and how to express it. That's at least partially why they started a company in the first place.

Some potential customers will be attracted to that personality and some will be repulsed. But that's true of anything that is wonderful and different and powerful in this world.

When a large company tries inject personality, it often comes off as contrived, not genuine. Whereas with a startup, it feels genuine because it *is* genuine.

Some customers only want to buy from the market leader. That's rational, and if that's a primary deciding factor, there's little a startup can do about it, no matter what their home page says. Therefore, the startup should ignore that segment and focus on customers who want to buy from a plucky upstart that has something to say.

Do things that aren't zero-sum

Some things in business are a zero-sum game.

In a zero-sum game, when one player wins, another player necessarily loses. Poker is an example: When one player wins chips, other players lose exactly that number of chips. Blackjack is a counter-example: Players at a table individually win or lose, unaffected by other players; no player loses chips when another player wins.

Marketing gives us examples of each. Zero-sum marketing channels arise when there's a power law or where there is exclusivity. Examples:

- SEO *(The top positions generate more traffic than all other positions combined)*
- AdWords *(The top positions generate more traffic than all other positions combined)*
- Affiliates *(The top few affiliates generate more leads than all others combined)*
- Retail shelf space *(Limited surface area)*
- Exclusive distribution deals *(The zero-sum game is created by agreement)*
- Government fiat *(A vendor can be written into law)*

Conversely, there are channels that are non-zero-sum, and more to the point, where even a well-funded, strongly-entrenched incumbent cannot prevent others from winning:

- Social media *(anyone can create a great social presence)*
- Newsletters *(anyone can create great content marketing)*
- Collaborative promotions *(both players more money than they would have)*
- Ecosystems like Salesforce and the Apple Store *(all players make more money)*
- Consumers who buy multiple products *(e.g. 3D animators often use multiple tools)*

Market dynamics can also create either type of game. Stagnant or shrinking markets are zero-sum; without new customers entering the arena, winning one customer means a competitor cannot have that revenue. In growing markets there's a steady stream of new dollars from new customers, so many competitors can grow. (This is yet another reason why startups should target growing markets, whether the goal is to build a small, profitable company or a unicorn.)

Incumbents are stronger in zero-sum games, because they can apply money and specialized expertise. They can even over-spend, because their scaled business model can absorb a low ROI activity, intentionally losing money in order to stop competitors from using that channel. This doesn't mean startups should *never* play zero-sum games, but they are more difficult, and sometimes impossible.

But incumbents cannot stop startups from winning non-zero-sum games, so that's where a startup should invest.

Be worse but acceptable in most dimensions

No one wants their website to go down.

It's surprising how hard it is to keep a website up for an entire year. For example, "99.9% uptime" might sound excellent, but that equates to 44 minutes of downtime every month! If our company WP Engine had even close to that much downtime, customers would revolt, and rightfully so. Yet, most hosting companies[7] promise only 99.9%−99.95% uptime.

[7] Including the major cloud providers, managed WordPress platforms like WP Engine, and specialized providers like Toasttab for restaurants

The reason is that it's very hard to push the number higher. It takes an order of magnitude more direct spend, software development, and processes to get to 99.99%, to say nothing of the proverbial "5-9's" that industrial operations sometimes target. Every little rare, strange, unpredictable thing will knock you out of compliance; it's expensive and difficult to solve all those cases. And yet, 99.99% just doesn't look that different from 99.95% on the pricing page.

Incumbents, however, often do have to invest in optimizing these expensive dimensions like. One reason is that at scale, rare things become common; at scale you have have no choice. Another reason is that it can win sales in some segments of the market; at WP Engine we have enterprise customers who have internal policies demanding 99.99% uptime, so we win against competitors with lesser guarantees.

Startups don't have the rare-at-scale problem, and they can choose their target market such that customers don't have extreme demands[8], and therefore startups can win while incumbents labor.

[8] Indeed, startups should actively avoid those customers. A startup with a new product definitionally won't satisfy customers with myriad, difficult demands.

"Kathy, if you agree to these terms of service, click 'I do.'"

More examples where incumbents have to care, but startups don't:

Security
WP Engine spends tens of millions of dollars a year on security, ranging from internal teams to corporate policies to annual employee training to SOC and ISO compliance to software reviews to vendor security reviews. For us, security is one of our main selling points, so this is important both for scale, brand, derisking, and because it's what customers pay us for. But that's not true of most products, and most customers don't demand it.

Quality
If the entire product is low quality in every dimension, that's just a bad product. I doubt anyone is excited to build that, not even as an SLC (MVP). For example, uptime is important for WP Engine, but for a SaaS product that is used only during normal working hours, targeted at a certain geography where "working hours" is a well-defined timeframe, having even an

hour of downtime in the middle of the night doesn't affect customers at all. A large, global company doesn't have that luxury.

Scalability

If a product will never need to handle "big data," then the product can be built with all sorts of simplifying assumptions[9] that make development faster, safer, even more enjoyable. The UX can be simpler if the users have basic needs, as opposed to nested security groups driven by an external LDAP service.

[9] PostgreSQL instead of BigQuery, SQLLite instead of PostgreSQL, Python instead of Erlang, reading into memory instead of streaming, batch jobs instead of assemblages of queues and auto-scale groups, standard algorithms rather than distributed computation, normal debuggers instead of distributed logs, off-the-shelf libraries and vendors instead of bespoke solutions

Compliance

Large companies accumulate internal policies. These are for good reasons, like ensuring that not everyone has access to all data (especially customer data), ensuring that IT teams are capable of managing and upgrading thousands of devices, safeguarding ownership of their intellectual property, and enabling sales to large enterprises and governments in various verticals who impose policies on their vendors. This is one of the reasons they can get multi-million-dollar three-year contracts and the start can't. But it also severely hampers what they can do, or at what speed they can do it, or at what cost, and therefore at what customer-facing price.

Legal

We covered this earlier. While I would never advocate for startups to do illegal things on purpose, it's a simple fact that startups often (unknowingly) don't comply with all laws. It doesn't affect their sales; in fact, it might increase it.

Most customers don't care about most things. This is great news for startups, who can select one or two dimensions to care about, and the ideal customer segment who also cares mostly about those specific things, and win that segment while incumbents chase complexity in all quarters.

While incumbents have to charge more to cover the costs of multi-dimensional excellence at scale, a startup can charge less for a product that's objectively "worse" along many dimensions, and thus the startup can win on price and still be profitable.

Startups can be worse, but unique, and better where it counts.

Be low-cost against the profit-center

An incumbent cannot change its business model.

The assets that give the incumbent its advantage are also static constraints. The brand is entrenched in consumers' minds. The software relies on platforms and languages and libraries that cannot be changed without a massive rewrite which infamously almost always fails. The business model of marketing, sales, service, and profits is set; a company with the costs of a global sales team, the white-glove on-boarding team, the expensive infrastructure, the vendor costs, and shareholder expectations that profits will only increase, cannot drop prices.

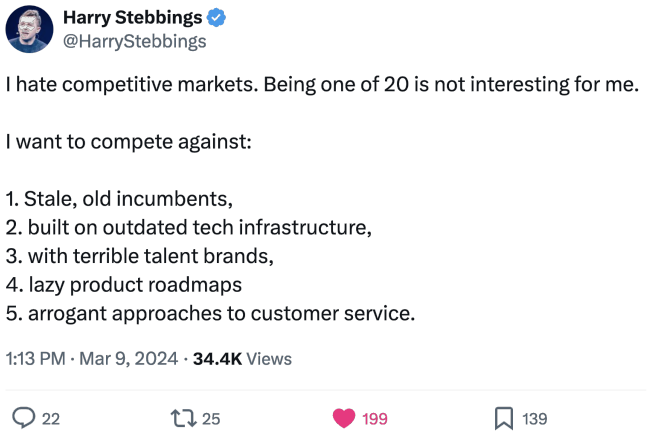
Therefore, an incumbent cannot compete with a startup which is "a 'lite' version for 1/10th the price."

This is a softer way of restating Disruption Theory[10], in which incumbents see the startup coming but, rather than compete directly, reposition themselves to focus on their best, most profitable customer segments, thereby allowing the startup to thrive.

[10] Famously explained in *Innovator's Dilemma*, the theory is more specific than what I'm saying here, involving new technology that is "worse, but cheaper, but in some ways better," where the incumbent seems to act rationally but ends up being completely disrupted.

More specifically: Whatever generates the most profits for the incumbent, is the thing they are least able to change.

You have to be careful, because incumbents will spend a lot of money and attention defending their profit centers from attack. But price won't be how they defend. That defense means you should leverage other topics from this article; for example an incumbent can decide to spend "too much" money on AdWords to make that a worse channel for you, but they cannot stop you from having a great content marketing strategy.



Incumbents are strong in most ways, but they are vulnerable.

The only mistake is for a startup to go head-to-head with an incumbent where the incumbent is strong.

Attack where you are strong, and they are weak.

This is how to build a winning strategy.

Printed from: *A Smart Bear*
https://longform.asmartbear.com/startup-beats-incumbent/
© 2007-2024 Jason Cohen @asmartbear