Stevey's Google Platforms Rant

24-31 minutes

I was at Amazon for about six and a half years, and now I've been at Google for that long. One thing that struck me immediately about the two companies -- an impression that has been reinforced almost daily -- is that Amazon does everything wrong, and Google does everything right. Sure, it's a sweeping generalization, but a surprisingly accurate one. It's pretty crazy. There are probably a hundred or even two hundred different ways you can compare the two companies, and Google is superior in all but three of them, if I recall correctly. I actually did a spreadsheet at one point but Legal wouldn't let me show it to anyone, even though recruiting loved it.

I mean, just to give you a very brief taste: Amazon's recruiting process is fundamentally flawed by having teams hire for themselves, so their hiring bar is incredibly inconsistent across teams, despite various efforts they've made to level it out. And their operations are a mess; they don't really have SREs and they make engineers pretty much do everything, which leaves almost no time for coding - though again this varies by group, so it's luck of the draw. They don't give a single shit about charity or helping the needy or community contributions or anything like that. Never comes up there, except maybe to laugh about it. Their facilities are dirt-smeared cube farms without a dime spent on decor or common meeting areas. Their pay and benefits suck, although much less so lately due to local competition from Google and Facebook. But they don't have any of our perks or extras -- they just try to match the offer-letter numbers, and that's the end of it. Their code base is a disaster, with no engineering standards whatsoever except what individual teams choose to put in place.

To be fair, they do have a nice versioned-library system that we really ought to emulate, and a nice publish-subscribe system that we also have no equivalent for. But for the most part they just have a bunch of crappy tools that read and write state machine information into relational databases. We wouldn't take most of it even if it were free.

I think the pubsub system and their library-shelf system were two out of the grand total of three things Amazon does better than google.

I guess you could make an argument that their bias for launching early and iterating like mad is also something they do well, but you can argue it either way. They prioritize launching early over everything else, including retention and engineering discipline and a bunch of other stuff that turns out to matter in the long run. So even though it's given them some competitive advantages in the marketplace, it's created enough other problems to make it something less than a slam-dunk.

But there's one thing they do really really well that pretty much makes up for ALL of their political, philosophical and technical screw-ups.

Jeff Bezos is an infamous micro-manager. He micro-manages every single pixel of Amazon's retail site. He hired Larry Tesler, Apple's Chief Scientist and probably the very most famous and respected human-computer interaction expert in the entire world, and then ignored every goddamn thing Larry said for three years until Larry finally -- wisely -- left the company. Larry would do these big usability studies and demonstrate beyond any shred of doubt that nobody can understand that frigging website, but Bezos just couldn't let go of those pixels, all those millions of semantics-packed pixels on the landing page. They were like millions of his own precious children. So they're all still there, and Larry is not.

Micro-managing isn't that third thing that Amazon does better than us, by the way. I mean, yeah, they micro-manage really well, but I wouldn't list it as a strength or anything. I'm just trying to set the context here, to help you understand what happened. We're talking about a guy who in all seriousness has said on many public occasions that people should be paying him to work at Amazon. He hands out little yellow stickies with his name on them, reminding people "who runs the company" when they disagree with him. The guy is a regular... well, Steve Jobs, I guess. Except without the fashion or design sense. Bezos is super smart; don't get me wrong. He just makes ordinary control freaks look like stoned hippies.

So one day Jeff Bezos issued a mandate. He's doing that all the time, of course, and people scramble like ants being pounded with a rubber mallet whenever it happens. But on one occasion -- back around 2002 I think, plus or minus a year -- he issued a mandate that was so out there, so huge and eye-bulgingly ponderous, that it made all of his other mandates look like unsolicited peer bonuses.

His Big Mandate went something along these lines:

1. All teams will henceforth expose their data and functionality through service interfaces.

2. Teams must communicate with each other through these interfaces.

3. There will be no other form of interprocess communication allowed: no direct linking, no direct reads of another team's data store, no shared-memory model, no back-doors whatsoever. The only communication allowed is via service interface calls over the network.

4. It doesn't matter what technology they use. HTTP, Corba, Pubsub, custom protocols -- doesn't matter. Bezos doesn't care.

5. All service interfaces, without exception, must be designed from the ground up to be externalizable. That is to say, the team must plan and design to be able to expose the interface to developers in the outside world. No exceptions.

6. Anyone who doesn't do this will be fired.

7. Thank you; have a nice day!

Ha, ha! You 150-odd ex-Amazon folks here will of course realize immediately that #7 was a little joke I threw in, because Bezos most definitely does not give a shit about your day.

#6, however, was quite real, so people went to work. Bezos assigned a couple of Chief Bulldogs to oversee the effort and ensure forward progress, headed up by Uber-Chief Bear Bulldog Rick Dalzell. Rick is an ex-Armgy Ranger, West Point Academy graduate, ex-boxer, ex-Chief Torturer slash CIO at Wal*Mart, and is a big genial scary man who used the word "hardened interface" a lot. Rick was a walking, talking hardened interface himself, so needless to say, everyone made LOTS of forward progress and made sure Rick knew about it.

Over the next couple of years, Amazon transformed internally into a service-oriented architecture. They learned a tremendous amount while effecting this transformation. There was lots of existing documentation and lore about SOAs, but at Amazon's vast scale it was about as useful as telling Indiana Jones to look both ways before crossing the street. Amazon's dev staff made a lot of discoveries along the way. A teeny tiny sampling of these discoveries included:

- pager escalation gets way harder, because a ticket might bounce through 20 service calls before the real owner is identified. If each bounce goes through a team with a 15-minute response time, it can be hours before the right team finally finds out, unless you build a lot of scaffolding and metrics and reporting.

- every single one of your peer teams suddenly becomes a potential DOS attacker. Nobody can make any real forward progress until very serious quotas and throttling are put in place in every single service.

- monitoring and QA are the same thing. You'd never think so until you try doing a big SOA. But when your service says "oh yes, I'm fine", it may well be the case that the only thing still functioning in the server is the little component that knows how to say "I'm fine, roger roger, over and out" in a cheery droid voice. In order to tell whether the service is actually responding, you have to make individual calls. The problem continues recursively until your monitoring is doing comprehensive semantics checking of your entire range of services and data, at which point it's indistinguishable from automated QA. So they're a continuum.

- if you have hundreds of services, and your code MUST communicate with other groups' code via these services, then you won't be able to find any of them without a service-discovery mechanism. And you can't have that without a service registration mechanism, which itself is another service. So Amazon has a universal service registry where you can find out reflectively (programmatically) about every service, what its APIs are, and also whether it is currently up, and where.

- debugging problems with someone else's code gets a LOT harder, and is basically impossible unless there is a universal standard way to run every service in a debuggable sandbox.

That's just a very small sample. There are dozens, maybe hundreds of individual learnings like these that Amazon had to discover organically. There were a lot of wacky ones around externalizing services, but not as many as you might think. Organizing into services taught teams not to trust each other in most of the same ways they're not supposed to trust external developers.

This effort was still underway when I left to join Google in mid-2005, but it was pretty far advanced. From the time Bezos issued his edict through the time I left, Amazon had transformed culturally into a company that thinks about everything in a services-first fashion. It is now fundamental to how they approach all designs, including internal designs for stuff that might never see the light of day externally.

At this point they don't even do it out of fear of being fired. I mean, they're still afraid of that; it's pretty much part of daily life there, working for the Dread Pirate Bezos and all. But they do services because they've come to understand that it's the Right Thing. There are without question pros and cons to the SOA approach, and some of the cons are pretty long. But overall it's the right thing because SOA-driven design enables Platforms.

That's what Bezos was up to with his edict, of course. He didn't (and doesn't) care even a tiny bit about the well-being of the teams, nor about what technologies they use, nor in fact any detail whatsoever about how they go about their business unless they happen to be screwing up. But Bezos realized long before the vast majority of Amazonians that Amazon needs to be a platform.

You wouldn't really think that an online bookstore needs to be an extensible, programmable platform. Would you?

Well, the first big thing Bezos realized is that the infrastructure they'd built for selling and shipping books and sundry could be transformed an excellent repurposable computing platform. So now they have the Amazon Elastic Compute Cloud, and the Amazon Elastic MapReduce, and the Amazon Relational Database Service, and a whole passel' o' other services browsable at aws.amazon.com. These services host the backends for some pretty successful companies, reddit being my personal favorite of the bunch.

The other big realization he had was that he can't always build the right thing. I think Larry Tesler might have struck some kind of chord in Bezos when he said his mom couldn't use the goddamn website. It's not even super clear whose mom he was talking about, and doesn't really matter, because nobody's mom can use the goddamn website. In fact I myself find the website disturbingly daunting, and I worked there for over half a decade. I've just learned to kinda defocus my eyes and concentrate on the million or so pixels near the center of the page above the fold.

I'm not really sure how Bezos came to this realization -- the insight that he can't build one product and have it be right for everyone. But it doesn't matter, because he gets it. There's actually a formal name for this phenomenon. It's called Accessibility, and it's the most important thing in the computing world.

The. Most. Important. Thing.

If you're sorta thinking, "huh? You mean like, blind and deaf people Accessibility?" then you're not alone, because I've come to understand that there are lots and LOTS of people just like you: people for whom this idea does not have the right Accessibility, so it hasn't been able to get through to you yet. It's not your fault for not understanding, any more than it would be your fault for being blind or deaf or motion-restricted or living with any other disability. When software -- or idea-ware for that matter -- fails to be accessible to anyone for any reason, it is the fault of the software or of the messaging of the idea. It is an Accessibility failure.

Like anything else big and important in life, Accessibility has an evil twin who, jilted by the unbalanced affection displayed by their parents in their youth, has grown into an equally powerful Arch-Nemesis (yes, there's more than one nemesis to accessibility) named Security. And boy howdy are the two ever at odds.

But I'll argue that Accessibility is actually more important than Security because dialing Accessibility to zero means you have no product at all, whereas dialing Security to zero can still get you a reasonably successful product such as the Playstation Network.

So yeah. In case you hadn't noticed, I could actually write a book on this topic. A fat one, filled with amusing anecdotes about ants and rubber mallets at companies I've worked at. But I will never get this little rant published, and you'll never get it read, unless I start to wrap up.

That one last thing that Google doesn't do well is Platforms. We don't understand platforms. We don't "get" platforms. Some of you do, but you are the minority. This has become painfully clear to me over the past six years. I was kind of hoping that competitive pressure from Microsoft and Amazon and more recently Facebook would make us wake up collectively and start doing universal services. Not in some sort of ad-hoc, half-assed way, but in more or less the same way Amazon did it: all at once, for real, no cheating, and treating it as our top priority from now on.

But no. No, it's like our tenth or eleventh priority. Or fifteenth, I don't know. It's pretty low. There are a few teams who treat the idea very seriously, but most teams either don't think about it all, ever, or only a small percentage of them think about it in a very small way.

It's a big stretch even to get most teams to offer a stubby service to get programmatic access to their data and computations. Most of them think they're building products. And a stubby service is a pretty pathetic service. Go back and look at that partial list of learnings from Amazon, and tell me which ones Stubby gives you out of the box. As far as I'm concerned, it's none of them. Stubby's great, but it's like parts when you need a car.

A product is useless without a platform, or more precisely and accurately, a platform-less product will always be replaced by an equivalent platform-ized product.

Google+ is a prime example of our complete failure to understand platforms from the very highest levels of executive leadership (hi Larry, Sergey, Eric, Vic, howdy howdy) down to the very lowest leaf workers (hey yo). We all don't get it. The Golden Rule of platforms is that you Eat Your Own Dogfood. The Google+ platform is a pathetic afterthought. We had no API at all at launch, and last I checked, we had one measly API call. One of the team members marched in and told me about it when they launched, and I asked: "So is it the Stalker API?" She got all glum and said "Yeah." I mean, I was joking, but no... the only API call we offer is to get someone's stream. So I guess the joke was on me.

Microsoft has known about the Dogfood rule for at least twenty years. It's been part of their culture for a whole generation now. You don't eat People Food and give your developers Dog Food. Doing that is simply robbing your long-term platform value for short-term successes. Platforms are all about long-term thinking.

Google+ is a knee-jerk reaction, a study in short-term thinking, predicated on the incorrect notion that Facebook is successful because they built a great product. But that's not why they are successful. Facebook is successful because they built an entire constellation of products by allowing other people to do the work. So Facebook is different for everyone. Some people spend all their time on Mafia Wars. Some spend all their time on Farmville. There are hundreds or maybe thousands of different high-quality time sinks available, so there's something there for everyone.

Our Google+ team took a look at the aftermarket and said: "Gosh, it looks like we need some games. Let's go contract someone to, um, write some games for us." Do you begin to see how incredibly wrong that thinking is now? The problem is that we are trying to predict what people want and deliver it for them.

You can't do that. Not really. Not reliably. There have been precious few people in the world, over the entire history of computing, who have been able to do it reliably. Steve Jobs was one of them. We don't have a Steve Jobs here. I'm sorry, but we don't.

Larry Tesler may have convinced Bezos that he was no Steve Jobs, but Bezos realized that he didn't need to be a Steve Jobs in order to provide everyone with the right products: interfaces and workflows that they liked and felt at ease with. He just needed to enable third-party developers to do it, and it would happen automatically.

I apologize to those (many) of you for whom all this stuff I'm saying is incredibly obvious, because yeah. It's incredibly frigging obvious. Except we're not doing it. We don't get Platforms, and we don't get Accessibility. The two are basically the same thing, because platforms solve accessibility. A platform is accessibility.

So yeah, Microsoft gets it. And you know as well as I do how surprising that is, because they don't "get" much of anything, really. But they understand platforms as a purely accidental outgrowth of having started life in the business of providing platforms. So they have thirty-plus years of learning in this space. And if you go to msdn.com, and spend some time browsing, and you've never seen it before, prepare to be amazed. Because it's staggeringly huge. They have thousands, and thousands, and THOUSANDS of API calls. They have a HUGE platform. Too big in fact, because they can't design for squat, but at least they're doing it.

Amazon gets it. Amazon's AWS (aws.amazon.com) is incredible. Just go look at it. Click around. It's embarrassing. We don't have any of that stuff.

Apple gets it, obviously. They've made some fundamentally non-open choices, particularly around their mobile platform. But they understand accessibility and they understand the power of third-party development and they eat their dogfood. And you know what? They make pretty good dogfood. Their APIs are a hell of a lot cleaner than Microsoft's, and have been since time immemorial.

Facebook gets it. That's what really worries me. That's what got me off my lazy butt to write this thing. I hate blogging. I hate... plussing, or whatever it's called when you do a massive rant in Google+ even though it's a terrible venue for it but you do it anyway because in the end you really do want Google to be successful. And I do! I mean, Facebook wants me there, and it'd be pretty easy to just go. But Google is home, so I'm insisting that we have this little family intervention, uncomfortable as it might be.

After you've marveled at the platform offerings of Microsoft and Amazon, and Facebook I guess (I didn't look because I didn't want to get too depressed), head over to developers.google.com and browse a little. Pretty big difference, eh? It's like what your fifth-grade nephew might mock up if he were doing an assignment to demonstrate what a big powerful platform company might be building if all they had, resource-wise, was one fifth grader.

Please don't get me wrong here -- I know for a fact that the dev-rel team has had to FIGHT to get even this much available externally. They're kicking ass as far as I'm concerned, because they DO get platforms, and they are struggling heroically to try to create one in an environment that is at best platform-apathetic, and at worst often openly hostile to the idea.

I'm just frankly describing what developers.google.com looks like to an outsider. It looks childish. Where's the Maps APIs in there for Christ's sake? Some of the things in there are labs projects. And the APIs for everything I clicked were... they were paltry. They were obviously dog food. Not even good organic stuff. Compared to our internal APIs it's all snouts and horse hooves.

And also don't get me wrong about Google+. They're far from the only offenders. This is a cultural thing. What we have going on internally is basically a war, with the underdog minority Platformers fighting a more or less losing battle against the Mighty Funded Confident Producters.

Any teams that have successfully internalized the notion that they should be externally programmable platforms from the ground up are underdogs -- Maps and Docs come to mind, and I know GMail is making overtures in that direction. But it's hard for them to get funding for it because it's not part of our culture. Maestro's funding is a feeble thing compared to the gargantuan Microsoft Office programming platform: it's a fluffy rabbit versus a T-Rex. The Docs team knows they'll never be competitive with Office until they can match its scripting facilities, but they're not getting any resource love. I mean, I assume they're not, given that Apps Script only works in Spreadsheet right now, and it doesn't even have keyboard shortcuts as part of its API. That team looks pretty unloved to me.

Ironically enough, Wave was a great platform, may they rest in peace. But making something a platform is not going to make you an instant success. A platform needs a killer app. Facebook -- that is, the stock service they offer with walls and friends and such -- is the killer app for the Facebook Platform. And it is a very serious mistake to conclude that the Facebook App could have been anywhere near as successful without the Facebook Platform.

You know how people are always saying Google is arrogant? I'm a Googler, so I get as irritated as you do when people say that. We're not arrogant, by and large. We're, like, 99% Arrogance-Free. I did start this post -- if you'll reach back into distant memory -- by describing Google as "doing everything right". We do mean well, and for the most part when people say we're arrogant it's because we didn't hire them, or they're unhappy with our policies, or something along those lines. They're inferring arrogance because it makes them feel better.

But when we take the stance that we know how to design the perfect product for everyone, and believe you me, I hear that a lot, then we're being fools. You can attribute it to arrogance, or naivete, or whatever -- it doesn't matter in the end, because it's foolishness. There IS no perfect product for everyone.

And so we wind up with a browser that doesn't let you set the default font size. Talk about an affront to Accessibility. I mean, as I get older I'm actually going blind. For real. I've been nearsighted all my life, and once you hit 40 years old you stop being able to see things up close. So font selection becomes this life-or-death thing: it can lock you out of the product completely. But the Chrome team is flat-out arrogant

here: they want to build a zero-configuration product, and they're quite brazen about it, and Fuck You if you're blind or deaf or whatever. Hit Ctrl-+ on every single page visit for the rest of your life.

It's not just them. It's everyone. The problem is that we're a Product Company through and through. We built a successful product with broad appeal -- our search, that is -- and that wild success has biased us.

Amazon was a product company too, so it took an out-of-band force to make Bezos understand the need for a platform. That force was their evaporating margins; he was cornered and had to think of a way out. But all he had was a bunch of engineers and all these computers... if only they could be monetized somehow... you can see how he arrived at AWS, in hindsight.

Microsoft started out as a platform, so they've just had lots of practice at it.

Facebook, though: they worry me. I'm no expert, but I'm pretty sure they started off as a Product and they rode that success pretty far. So I'm not sure exactly how they made the transition to a platform. It was a relatively long time ago, since they had to be a platform before (now very old) things like Mafia Wars could come along.

Maybe they just looked at us and asked: "How can we beat Google? What are they missing?"

The problem we face is pretty huge, because it will take a dramatic cultural change in order for us to start catching up. We don't do internal service-oriented platforms, and we just as equally don't do external ones. This means that the "not getting it" is endemic across the company: the PMs don't get it, the engineers don't get it, the product teams don't get it, nobody gets it. Even if individuals do, even if YOU do, it doesn't matter one bit unless we're treating it as an all-hands-on-deck emergency. We can't keep launching products and pretending we'll turn them into magical beautiful extensible platforms later. We've tried that and it's not working.

The Golden Rule of Platforms, "Eat Your Own Dogfood", can be rephrased as "Start with a Platform, and Then Use it for Everything." You can't just bolt it on later. Certainly not easily at any rate -- ask anyone who worked on platformizing MS Office. Or anyone who worked on platformizing Amazon. If you delay it, it'll be ten times as much work as just doing it correctly up front. You can't cheat. You can't have secret back doors for internal apps to get special priority access, not for ANY reason. You need to solve the hard problems up front.

I'm not saying it's too late for us, but the longer we wait, the closer we get to being Too Late.

I honestly don't know how to wrap this up. I've said pretty much everything I came here to say today. This post has been six years in the making. I'm sorry if I wasn't gentle enough, or if I misrepresented some product or team or person, or if we're actually doing LOTS of platform stuff and it just so happens that I and everyone I ever talk to has just never heard about it. I'm sorry.

But we've gotta start doing this right.

Original URL

Follow-up URL

Hacker News URL