EPISODE 17

[INTERVIEW]

[0:00:00.3] JM: Ilya Sukhar, welcome to Software Engineering Daily.

[0:00:02.8] IS: Thanks for having me.

[0:00:04.6] JM: You were the CEO and founder of Parse. Explain what Parse was.

[0:00:09.6] IS: Parse was a company that set out to make mobile development dramatically easier for the set of developers that were in the ecosystem in 2011. We grew to generally tackle development and trying – I want to just start over.

[0:00:26.3] JM: Sure, sure. Go for it.

[0:00:28.0] IS: It's been a while since I've given the elevator pitch.

[0:00:30.0] JM: Get a little closer to the mic. You want to be that –

[0:00:38.3] IS: Parse in a sentence, made cloud services for mobile developers. We wanted to make it dramatically easier to spin up an app and get all the basic functionality, particularly on the backend that every app needed at the time almost for free; cloud storage, user accounts, analytics, push notifications, backend logic. We were trying to bridge the gap between the types of folks that were building the new client-heavy native mobile apps in 2011 when we got started and the typical tools that people had on the backend, Rails, Django, whatnot, that were not really oriented to the challenges of the time. They didn't really integrate with the mobile SDKs, Objective-C, the Android stack. We're trying to bridge that gap with a set of cloud services and a set of heavy client SDKs.

[0:01:38.8] JM: How did Parse compare to the other backend-as-a-service products that we've seen over time? When we think about the Heroku, Firebase, more recently Zeit perhaps. How does Parse fit into that set of products?

[0:01:55.8] IS: I think among those, we were the first to be mobile first and oriented around mobile. Our North Star was you're an Objective-C developer, you’re focused say on iOS, which is where we get started. You think mostly in terms of the front-end, how do you very quickly plug in the backend? Whereas, say Heroku for example, was oriented around hosting Rails apps, hosting Django apps. The premise was you're already building that app, you're in the Rails environment, but this is how to deploy it, how to make it reliable, scalable and whatnot.

I think we were much more full-stack, I would say and we were much more opinionated about how to do things. We had our own closed ecosystem, our own universe so to speak. We were very focused on mobile as the entry point. We generalized beyond that as we grew, but I think none of the others started there. Some of them shifted their focus to mobile over time, but we started there.

[0:03:06.3] JM: What were the difficulties of building on AWS back then? This was, I think what? 2011?

[0:03:13.6] IS: 2011. Yeah. Well, it depends who you are, but we were building for these mobile developers. If you're a front-end oriented mobile developer – Why are you smiling?

[0:03:29.2] JM: I just smile. That's what an interviewer is supposed to do when they’re giving off nonverbal cues of acceptance and understanding.

[0:03:36.8] IS: Okay, fair enough. Imagine you're a mobile developer and you're primarily trying to build, say a game. Most of your challenges at the time are how do I build a beautiful app? How do I get the UX right? How do I do the transitions correctly? This is 2011, so mobile is somewhat new. How do I really orient to this new ecosystem?

The backend for most of the apps at the time and I would argue for most of the apps in the consumer app store today is pretty simple. You think in terms of how do I store and retrieve structured data? How do I send out push notifications to my users? How do I do user accounts when I start data against those user accounts?

I would argue our lens on this was AWS is just like an alien universe, right? You have to think about EC2, like an actual Linux machine. You have to spin up a database. You have to figure out how to make it reliable and scalable. You have to figure out all of the middleware, which at the time was getting, I guess plugged by projects like Rails. At the time, those projects had no strong opinions about how they should integrate with mobile. There was no Objective-C SDK that would basically handle the networking, the caching, the object model, the ORM in the client, tie it back to the backend. You had to interface with a bunch of stuff that really wasn't your core competency.

AWS was fine for people who are coming from okay, how do I apply my Rails app that interfaces well with clients that are web browsers? If you were starting from mobile and backend was a means to an end, not your primary differentiator, I would argue it’s just a alien universe that you didn't want to deal with.

[0:05:34.9] JM: Was that the hardest set of engineering problems that you had to deal with was this – the full stack developer experience, integration, API service surface? Or was it – were there also backend scalability problems that exhibited more acute challenges?

[0:05:57.8] IS: On our end?

[0:05:58.4] JM: Yeah.

[0:05:59.5] IS: Oh, yeah. I mean, it's all of the above. I think we had to design an elegant system that abstracted away all of these challenges, the startup challenges from developers, all the maintenance challenges from developers, or customers. Yeah, we ourselves definitely dealt with a lot of scaling challenges. When we sold the company to Facebook, we had somewhere between say 60,000 and 80,000 active developers. Through our height at Facebook, it went a couple of order – was a magnitude higher than that.

We definitely had a lot of challenges of scaling, dealing with all the workloads on the platform. We for better or for worse had – Mongo is the primary data store underneath everything. I think that was great because the Mongo data model exposed itself well in the various higher layers of our stack. At the end of the day, Mongo was not an easy thing to scale, not an easy thing to isolate customer’s want from another, not an easy thing to manage massive indexes. We definitely had to innovate there and in some cases, fail.

[0:07:19.5] JM: Like what? What's hard about managing that?

[0:07:27.2] IS: We had the extreme version of multi-tenancy, right? We were effectively munging together tens of thousands of dollars on instances of Mongo, where Mongo was not designed for that. We had a back-end runtime that looked a lot like node, where basically you could write snippets of backend logic, business logic that would integrate into our system. You would get all the benefits of the integration with the client SDKs. You get all the benefits of integrating with the object model. You would get baked in stuff, like semi-native Twilio support or whatnot and you're just writing literally little snippets of functions at a time.

This was a custom runtime that we built off of V8, because node was very nascent when we got started and node was not really oriented around the multi-tenancy we needed. That was challenging, right? It's like you have an open runtime that anyone can run anything on. How do you put boundaries on that? How do you scale that? How do you ensure that people don't mind crypto on it, or don't – we have some guy index try to pull FCC documents off government websites, using us as a crawler. I became friends with them at some point, but I was like “Dude, what are you doing?” We were just an open platform, an open runtime. At every level of the stack, we had to face some pretty serious scaling challenges.

[0:09:01.8] JM: Can you go deeper into the custom runtime built off of V8?

[0:09:08.1] IS: Yeah.

[0:09:10.6] JM: Why did you need to build that and what did that consist of?

[0:09:13.7] IS: Sure. Maybe I'll give you the rough sequence. I'll probably won't remember this perfectly. The rough sequence of how we built the product. The very first version of the product was an iOS SDK, where iOS developers can store and retrieve structure data. Basically, we'd serialize NS dictionaries, which is a core object in Objective-C. serialize it down all the way down. so you wouldn't have to think about the network layer. you wouldn't think about the caching. you wouldn't think about the database. you wouldn't have to set up your schema ahead of time. so you could declare an object in Objective-C. We would infer the schema. set up your tables for you. set up indexes for you. It's pretty cool. That was V1.

On top of that, we added things like the notion of user accounts, permissions, fast-forwarding, push notifications, some analytics. A the end of the day, you don't want to do everything in the front-end oriented. You need some business logic to run on your backend, write the type of code you would put into a Rails controller, let's say. Where does that live? We didn't have a place for that, right? They say, you're an iOS game developer, you wanted to have a high scores list. At some point, there needs to be some back-end code that actually sorts and sends out the high score list. Otherwise ,every client would be relying on some raw version of it in the database. They could write to it any way they wanted.

We had to build that layer. What we did was we built a custom, called the cloud code. In today's parlance, it would be a serverless environment, where you don't think about servers, you don't think about scaling, yet you're writing little snippets of JavaScript that interact with our object model, that interact with our SDKs. You can do things, like in that case, compute the high scores list, push it back up to the client, or do something with data, send out an e-mail, do something with data, send out a text message, that kind of thing.

[0:11:41.0] JM: Were there any strategic decisions that you encountered at Parse that were particularly strange or unique? More at the business layer. You've just given a good example of a hard engineering problem. You've had to solve them. I'm wondering about the business and strategic decisions that you had to make and what was unconventional.

[0:12:08.9] IS: Unconventional. Some of these things I think are not unconventional with today's eyes. I think back then, maybe they were unconventional. We had a very bottoms up approach to things. We shipped a very bare-bones product to start. I think we differentiated by having a really keen ear for the customer iterating quickly and over-delivering on support and over-delivering on really the value you got for the price you paid. We had a very generous free tier.

The vast majority of folks never paid us anything. I think that got us a widespread brand. It got us a lot of developer love. It really got us a lot of projects that got started on Parse. I mean, even today, every week I meet people who are like, “I don't use Parse anymore, because I can't, but I got started on Parse. My startup exists because Parse was there at the right time.”

This presented a number of challenges as we grew. One is it's just very hard to provide the same level of service as you grow, so it's easy to answer the first few hundred e-mails that are asking for help and some of the help is more or less debugging their app, their code, as opposed to their integration with our code, over-delivered on that stuff. At a certain point, you need to make money. I think we were not the best at squeezing out all the value we could, but we did an okay job at this. I think in our pre-Facebook days, we made money off of a couple of types of apps. One was game developers, so I start there and I don't think a lot of people know this, but it's a pretty vibrant scene on the mobile app stores in those years, of smallish game developers making decent money, getting to real scale, lots of one-hit wonders, maybe to some extent.

We made a fair amount of money catering to them, because they had pretty simple back-end needs. They were staffed full of folks that were very client-oriented. There was a lot of product market fit there. We also made a good amount of money catering to creative agencies that we're building these fairly thin apps for their clients, so places like Home Depot, or the NFL, or my favorite was Sesame Street. They were very polished, nice apps. They weren't complex logically. You could argue that some of them weren't used that widely, although some were. The Sesame Street app was popular.

Again, we did a good job catering to them because there's a good product market fit that their backend needs were not that complex. Their scale wasn't that high, but they really need to move quickly. The more stuff they could not do that wasn’t core to their value prop to their clients, the better. We were able to give them a bunch of niceties that traditional backends wouldn't. We had a nice GUI. I forgot to talk about this, but a nice GUI interface for interacting with all of your data, your database, your push notification system, your analytics. Teams in the agency are on the client side. The product manager could log in and change some strings around, could send out some push notifications, could change the price of an item, that thing, in a way that didn't require going into a shell and dropping down to a real database.

Lastly, we had a fair amount of startups that got started and got to some real scale. Nonetheless, we definitely struggled with the strategy, the pricing around being this bottoms up, anyone can get started thing. How to keep the service from a reliability infrastructure perspective great, from customer support thing great, customer support angle great, while not jacking up the prices on people to the extent that they wouldn't get started on Parse. They would go through the slog of spinning up AWS, because it was cheaper.

[0:16:50.1] JM: When did the talks of acquisition begin and what was that like? I mean, you must have been fielding acquisition offers from different people. Tell me about managing the different offers and how you balanced that with operating the actual company.

[0:17:18.1] IS: Yeah. I think the acquisition talks came in various flavors, at various times with Facebook in particular. We actually had two rounds of discussions. The latter round converted obviously. The first round came probably a year prior and was different. The first time around, they were interested in us, I think only for the talent. The set of folks on the team who I think at the time were pretty world-class and mobile development and the architecture around mobile and its interface with typical backend systems. They were going through their own mobile transition and you've talked to many of the relevant folks on your podcast, who many of them are not my friends, but I met through Facebook.

The first time around, it was basically like, what you've got here is cool. We don't really care about it. We care about is what you've learned along the way, what you know about mobile developers in the ecosystem and how they're doing things, you guys have architected things, come work on our mobile transition. I have to say that was cool to have that conversation. It was cool to meet everyone and Brett Taylor still in the company and we met Zuck through that. We met a bunch of people.

We said no, because we were excited about we were doing. We were growing very quickly. I think at the time, we had just raised our series A, or we were just about to. One way or another, things were going fine. We're just excited about we were doing. A lot of these challenges that I talk about hadn't really come to a head.

There were a number of companies that around the same time, a little bit later, a little bit earlier, came around with roughly the same proposition, which is guys had built a good team, you seem to know what you're doing in this mobile ecosystem, it's all nascent and maybe we don't have as many of those people come join our team. This was a somewhat common pattern in Silicon Valley, I think at the time. I think it's much less common these days, because I think companies are having less of a hard time recruiting. I think the acute pain points around certain technologies are no longer there. Although, arguably maybe it's still happening in machine learning, and so to speak, AI.

Anyway, so this was a pattern. We talked to just about every big tech company, but we were not interested in that. The Facebook thing came around again much later and it took a different form. We're like, "What you've continued to do here?" We have a bunch of strategic questions on our end and our platform business, come more or less continue doing what you're doing, keep the brand, to keep the service up, keep your team together, let's build an AWS-like business with your brand, your starting point, your mobile focus. It's not going to be quite as expansive as AWS, but let's start there and leverage Facebook's data centers capital, people, brand, all those things, to do it better.

Obviously, it didn't come to fruition, but that was super exciting. I mean, we were always hoping to build something like AWS just at a different abstraction level, with a different ethos, with a different focus. Yeah, it was hard as an independent company, especially at that time. The venture funding market for what we were doing was not terrible, but not as good as it is today. People are very dubious about the value proposition that developer experience matters, that the whole look and feel the product matters, that all the integration points matter, that API design is a differentiator, that you can build services on top of the basic building blocks that AWS and others offer and extract margin off of that.

There's lots of companies we can talk about that are doing that today, have raised tons of money, doing fine. At the time when we were fundraising, we did okay and we could have raised another round. In fact at the time, we had Series B term sheets when we sold the company, but the venture market was in short, not really having it. They wanted to push us in a different direction. They wouldn't give us as much money as we wanted. When we had these conversation with Facebook it's like, "Oh, that's really exciting and it could be a huge accelerator." It was for a couple of years. Maybe onto the next question, but the proposition at the time was really cool.

[0:22:13.6] JM: To move quickly towards something I'm really curious about, why didn't the acquisition work out in the ideal way that it could have potentially manifested then?

[0:22:38.1] IS: Yeah. I wish I had a perfect answer here. I think it would have saved me a lot of stress and sleepless nights. I think fundamentally, the company moved on. It's hard to say exactly when, but I think all of the – let's say underpinnings of doing the deal and investing resources. I had counted all that into Parse. I think somewhat became irrelevant over the time we were there.

To set the context and this is from my point of view. I think if you asked other people, they might have different answers. I don't think there's any one truth here. From my point of view, when we sold the company the platform team had a little bit of an – the platform team at Facebook had a little bit of an existential situation on their hands. The whole company had not yet done the transition to mobile. The mobile apps were getting better, but when we got there, there were no ads in the newsfeed at all on mobile. That was a big question in the stock market. The company had IPO'd and the stock was down a ton. At the time of the deal, I think it was at 25.

There's a big strategic question overall of is Facebook going to make the transition to mobile? Then the platform team specifically, this is rewinding a while, but platform team the real way it contributed core value to the company was through the desktop gaming platform that Facebook powered, right? Zynga, all the generation of flash games that people are playing on Facebook. I don't know if you remember that. Yeah. They made a lot of money off of that, right? Because they had an app store model. They would provide the distribution and the –

[0:24:37.0] JM: This included Farmville, right?

[0:24:38.9] IS: Yeah. I don't know what the breakdown was between Farmville and the rest of the ecosystem. I'm sure there were other contributors, but Farmville was a big one. I think Candy Crush was on there. There's a lot of serious traffic going through that, and so they were taking attacks much like Apple and Google do these days on the payments for virtual goods that were flowing through the platform. That was not going to transition to mobile because Apple and Google control the app stores there and they control the distribution, they control the payments there. That business, which I don't remember what percentage of the business it was, but it was really not insignificant. It was a big chunk of the overall revenue company.

That business was going to do fine for years, but it wasn't going to grow, because everything was transitioning to mobile. It was obviously on a glide path down, because you can't redo that in the mobile ecosystem. To recap, the company had an overall strategic question of are we going to make the transition to mobile? The platform team had an overall question of what are we going to do? We were one of the experiments, I think. At least this was how they pitched it to us was we have all these developers, we have obviously data centers, we have technology that's really interesting. How can we build a different developer-oriented business that will in our case monetize infrastructure, APIs, databases, that thing?

They had other experiments. Some of which did really well. The app install ads that you probably see in your Facebook newsfeed, those killed it and they killed it with a small team and they'd create. Two or three years into the experiment, obviously the company had made them all the transition and then some overall, right? Stock was up a ton, printing money, newsfeed working great, newsfeed ads working great, app install ads working great for the platform team.

No matter how well we did, no matter how much developer love we had, how many apps we had, how many users of those apps we had, how much revenue we were making at the end of the day, it's really hard to compete with a really well-oiled ad business. Ad businesses are very efficient. Very, very efficient. Some of the conversations I had with Zuck, with my kernel, with other people, I think no one really wanted to say it outright. At the end of the day, they could give us an engineer to toil away on scaling Mongo and transitioning Mongo to Tao or any number of really cool engineering projects we had to make our service better.

That engineer put toward ad targeting, put toward the core business at that time was just a way better deal. I can't really argue with that. Maybe other people have different explanations for why it didn't quite work. Maybe our execution could have been better. Maybe Facebook's heart wasn't ever in it. I think the most fundamental explanation is that to yours into it let's say. Facebook didn't really need Parse.

[0:28:02.3] JM: Okay. Business-wise, that's obviously valid. Let's go all in on the thing that's working and let's just double down on that, because ads business is so good. This is not a direct mapping, but I sometimes think about Amazon fresh and how Amazon just never gives up on Amazon fresh. Maybe Amazon fresh is a great business, I have no idea. To me, it seems like they've always struggled with that business, but they've always kept up the long-term view that someday we're going to need to do groceries and we need to have this ongoing thing that's the grocery experiment. We're just going to allocate resources continually to it.

I mean, today Facebook has an entire team working on React. Is it something about just the expected value of having an ads business working that they were starting to see where they were like, "We need to go all in on this. We can't even spare five to 10 engineers to maintain this developer platform." When you think about it from a CEO's eye view, do you think it was the right capital allocation move to make, or you just think it was a valid capital allocation and move to make?

[0:29:22.8] IS: There's a lot here. To answer your concrete question, I think it was a valid allocation to make. I don't think it was long-term the right one. I think people don't quite appreciate that Facebook is a very different company than Amazon, or say Google. Both of which I think are much better at having these skunkworks, give 20 engineers to a team over there and leave them alone for a couple of years projects.

Facebook doesn't really have a good history of doing that. Someone once described this to me. I don't know if it totally holds water, but Facebook has this core rocket of this news feed. The newsfeed is a very flexible system. I mean, maybe this is less relevant in today's world, but certainly at the time, newsfeed is a very flexible system growing quickly. You can put any number of experiments into that architecture and see what users like and what sells ads.

I think Facebook made a key decision of let's try things. If they don't take off quickly, let's walk away from them. I think Facebook's core strength, Zuck's management orientation, certainly at the time I was there, is every six months let's have a few priorities. Let's reorient the company around those priorities. Let's shuffle teams. Let's blow up the world. Let's leave a lot of messes behind us, but let's always be super focused on those things and be very intellectually honest about where the leverage is in the system.

I think there's a lot of advantages to that. It's worked. It's hard to argue with. I don't think the company is that good at having these innovative off to the side teams that don't fit on the same timelines, don't fit on the same user expectations, don't have the same revenue model. It's just different.

[0:31:45.1] JM: During that time when you were starting to see the writing on the wall, were you able to depersonalize that process of Parse getting moved to a lesser priority within the company, or did it hurt? Do you feel it personally?

[0:32:06.7] IS: It hurt a lot. Now I think I have a pretty detached view of it. I think I'm comfortable with everything. At the end of the day, that whole thing did change my life. It changed the life of my co-founders. It changed the life of a lot of our employees who all did really well. We worked on cool stuff. It's hard to really be upset about it. At the time, yeah, it was just brutally painful. Because just if you took us in isolation, if you took our growth, how much developers loved us, how much they were using us, how much we had more and more examples of apps growing up on the platform and sticking with the platform and doing well on the platform. If you had lobbed us off, just cut us off and made us an independent company again, I think would be a very strong startup.

It's really painful to just have it all be put into this relative internal market of priorities at Facebook and realize that, “Yeah, maybe it's not the highest priority, but it's the thing that I really loved working on, the thing that I was personally enmeshed with.” Yeah, it was painful. It was tough for me.

[0:33:23.7] JM: Did you push to try to get them to change their minds, or did you just go along with it?

[0:33:32.2] IS: Yeah, totally. I mean, I think I took the brunt of the battles to keep it top of mind, to get it resourced correctly, to keep it going. It's all gradual, right? Maybe in the first six months, there were inklings of weird missed set expectations in terms of what folks were asking me and the rest of the team to do, that didn't seem to align with the core premise of the acquisition. For the first couple of years, I would say we were pretty well-resourced and we were left alone.

We were shipping stuff. We were growing quickly. We were presenting our product at F8. We were definitely benefiting from the scale of Facebook. I was in charge of a lot of other platform products at Facebook and I was able to bridge that gap in a lot of ways. I think we were helping some of the platform efforts pretty meaningfully with our expertise, in terms of building good APIs and thinking about stability and backwards compatibility and just generally rubbing off while I think on the rest of the platform organization of Facebook.

At a certain point, we weren't getting as many engineers as we wanted. It was obvious that Zuck's mind had moved on to Oculus and WhatsApp. A lot of our efforts to transition some of our technology to a better version powered by Facebook core technology, so what would Parse be like if the fundamental underlying data model in our underlying database was not Mongo, but it was Facebook's TAO. What would it be like if we weren't tied to our super custom version of effectively a node runtime, but we take advantage of all the cool container stuff going on inside the company, outside the company and offer that cool serverless thing in any language, or maybe a broader set of languages?

There are a bunch of cool stuff happening. What if we could really move to Facebook datacenters and change the economics of how we charged for things because we weren't paying the AWS tax? Lots of cool stuff was happening, but became pretty clear that people are less and less and less excited about it. There wasn't any one moment. I would say, you can ask anyone on the team, I fought pretty hard until I burned out.

[0:36:15.8] JM: Yeah. I still want that. Can you go back to Facebook? Can you like, "Mark, let's put the band back together."

[0:36:23.7] IS: It's funny. I don't think I would go back to Facebook, but I do think there is a need for something like Parse in today's ecosystem, because I think if you look at from the perspective of a client-heavy development team that just needs the basic fundamentals of crud backends, or even more complicated things than that, I don't know, I think we've back slid from certainly the days of Parse and we've definitely back slid from the days of Rails and Django, I think.

The React ecosystem has done amazing wonders for JavaScript on the client side, but I think the node backend ecosystem is super fragmented. There isn't this Rails omakase view of here's how you do it up and down the stack and it all works nicely together and you define your model in one place and it has all these ripple effects and it just happens. It feels, like when I spin up a side project today, it feels I'm making a dozen architectural decisions about stuff that doesn't really matter. None of my projects differentiate on how they lay out their schema in the database, or how they do migrations in the database, or how they do serialization to and from the client.

The average developer has to think about that still today and I think it's crazy, and they have to think about it across a bunch of these slices up and down the stack that are changing all the time. Everyone's got their own next new thing. They're not coordinated around any particular philosophy or taste, or API design. I do think that something is necessary. Yeah, I'll leave it there.

[0:38:14.9] JM: Firebase was started I think the same year as Parse?

[0:38:18.9] IS: Yeah.

[0:38:21.4] JM: Were you watching Firebase out of the corner of your eye, or was their service very different than Parse back then? Because now, that's like –they're very strongly branded as a backend as a service.

[0:38:33.8] IS: Totally. Yeah.

[0:38:35.2] JM: I think to some extent, they do fulfill some of the omakase stuff that you're describing.

[0:38:40.9] IS: That's fair. Yeah, I think they've done a really nice job there.

[0:38:42.7] JM: At least today.

[0:38:43.3] IS: Yeah, totally. It's funny, a lot of well, a fair amount of ex-Parse people actually work on Firebase at Google. Yeah. There's been a cultural intermingling there. I certainly wouldn't take credit for anything Firebase is doing. I think those guys have done a really good job there. Philosophically, I think they converged in a lot of ways, which is great. I do think they've done a good job.

Back in the day, it was pretty different. First of all, I think those guys were working on something entirely different in our YC batch. Then they had I think a similar, maybe philosophical approach to what they wanted to do, but we were laser-focused on mobile, native mobile, like the heavy clients that were just coming to the forefront of things. People forget that there was a whole wave of HTML5 Genki mobile stuff where people tried to build this halfway point between the web world and the mobile world. We're focused on that.

The Firebase guys as I recall, were really focused on real-time web. They really thought that web apps were going to get much more real-time and that they were going to power this architectural revolution in that ecosystem. They're very web-browser focused, very focused on I don't exactly remember how it worked, but holding open sockets and doing live updates on queries, really focused on lightning fast passing, message passing, so you could build real-time chat and other experiences, real-time gaming in a web world, which I think was really cool.

I don't think the web actually turned much more real-time than it was before. don't think the breadth of things that benefited from all that real-time stuff was that large, but they were on the same philosophy, and I think over time moved back around to more of a generic architecture, more of a mobile-oriented architecture. Yeah, that things did converge once we were at Facebook, they were at Google, I think more or less converged the same ethos.

[0:41:02.7] JM: I don't want the entire – we won't spend that much more time on the Parse Facebook stuff. I am a little bit curious about the winding down process, because I think it got turned into an open source project, but it's still a hosted thing. Then you eventually left. Can you just tell me about the winding down and how you figured out the best plan to go through that process?

[0:41:36.8] IS: Yeah. I will say, I left once this ball got rolling. I was quite burned out and I didn't want to expend more energy managing it down. My co-founder, Kevin, did a really remarkable job leading that effort. In the end, I think it was the cleanest way to do it. Basically, he built a open source version of Parse, which was not really the – it's not really open sourced the fundamental true codebase, because by then it had become pretty entangled with there's Facebook services. It was never designed to be open source. It just wasn't really possible to truly-truly open source it.

He did a good job of basically spinning up a clean room version of Parse that could be open sourced. They worked on it for almost a year, I think. In that time, gave notice, told developers what was going to happen, tried to work – actually successfully worked with a number of cloud providers to have a hosted version of the open source thing available, so that people can migrate on to. I think a bunch of cloud providers, AWS, Microsoft, others, smaller ones too were pretty eager to cooperate there, because it was free flow of developers. It had been pretty significant for print at that point.

Yeah, that's what happened. They did a pretty good job of it. In the end, it was painful for sure, but it was a pretty lengthy transition, about a year. The open source project is actually pretty vibrant still. None of the core Parse people are involved, but there's a site of people who had built their businesses on Parse and who were still invested in it and wanted to keep that going. There's maintainers and there's releases and there's a whole thing.

[0:43:53.8] JM: You've been part of three acquisitions. What general reflections do you have about software company acquisitions? What makes an acquisition successful, or unsuccessful? I mean, apart from the Facebook one. I think that was pretty interesting edge case that you described there, or maybe that's not edge case. I don't know. You could just tell me what your reflections on acquisitions are.

[0:44:27.0] IS: Yeah.

[0:44:28.8] JM: Software company acquisitions.

[0:44:31.8] IS: Yeah. I think they are more likely to succeed when the distance between the acquirer's product and the acquiree's product is minimal when the business model distance is minimal. You look at something like Instagram. We're probably a year behind Instagram, so we did get to see a lot of their journey at Facebook. Obviously, that worked tremendously well. Credit to those teams and certainly, you don't want to discount that. At the end of the day, it's a social product. It had a feed. It had all the same challenges, all the same leverage points, all the same organizational needs.

It was more about a question of did users love the product, as opposed to the challenges of integrating teams and the challenges of different cultures and the challenges of different demands and business models and all that stuff. I do think you either need to acquire stuff that you can plug in pretty quickly and give a lot of leverage to and just fit into the mold, or you need to acquire stuff and really let it alone for years and have a very clear understanding of what are the KPIs, what is success, and have a minimal interface between the organizations, like a good API design, right? Minimal surface area, very clear contracts. Give people all the room to have their own complexity internally.

I don't know if there's that many examples of that working in the ecosystem. I do think the average acquisition doesn't really bear that much fruit. There's obviously great standouts, but it's tough. One thing I totally, totally under-appreciated is just the machinations of a big company, right? When you're selling your company as a CEO, you hear the strategy as it is in that moment and you are interfacing with people in power in that moment. The reality is companies go through reorgs, people leave the company, the company shifts strategy. I think, acquired startups are often roadkill in those changes. I think unless you've worked in a big company, in a fairly senior position, you probably don't know what you're getting into when you sell your company.

I certainly had not worked at a big company in a senior position. I mostly worked at startups prior. When I talk to people these days, and this happens quite a bit, about whether or not they should sell their company, I really recommend they figure out which parts of the underlying premise behind acquisition are enduring and which could frankly change in three months based on whether the VP sponsor hits his numbers, leaves the company, gets battlefield promoted to a different priority. They could just be roadkill in that transition. It is a small set of things that matter at the end of the day.

[0:48:18.4] JM: Facebook is not the only large, scalable, highly distinct culture that you've become pretty aware of.

[0:48:31.3] IS: To be clear, I like Facebook in many ways.

[0:48:33.4] JM: Oh, yeah.

[0:48:33.9] IS: I had a good time there. I've made a ton of friends there. The economic impact of our acquisition was tremendous. I don't want to sound like I –

[0:48:43.3] JM: I don't think you come off that way. You also spend a lot of time with Y Combinator. Y Combinator is interesting, because it's – I think people don't – I don't know much about the internal culture at Y Combinator. It's clearly very strong and very distinct, but Y Combinator is an organization that scales. They've scaled. They will continue to scale. Their opportunities for scalability are tremendous. It's a different model for scalability it seems. I can't really map Y Combinator to the innovations of Facebook, or the innovations of Google, or Amazon, or something.

When you think about scalable innovation, what reflections do you have on Y Combinator? Perhaps, even in in contrast, or in comparison to the other innovation models that were more familiar with like the Facebook model, which you've contrasted with the Amazon, or the Google model.

[0:49:57.1] IS: That's an interesting question. I mean, I think the best analog for what YC does is higher education, right? I think they have built a remarkable brand around accelerating already very high potential people and teams and visions have developed a curriculum around what irrespective of whether you're a biology company, or a battery company, or just a regular old SaaS company, they've developed a curriculum of standard practices, best practices, startup wisdom, mentorship that generalizes. They do that twice a year with many hundreds of people.

I think the thing that I think maybe is counterintuitive about it is that it does scale in that – there is this general set of principles, this general way of doing things, general way of thinking about growth in your company, how to measure yourself in that early stage, how to talk to investors. These fundamental startup principles that any partner at YC after a little bit internalizes and starts to become part of the hive mind and in view into all of the startups that they work with.

It is unique in the sense that they don't necessarily need to innovate the organization that much. I mean, they have done a lot of interesting things, they've gone later stage, they've added research arm, this and that. The core of what YC does feels very much the same as it did in the Paul Graham era when it felt more of a family business. It’s just that his worldview and his approach to all of this turns out to be pretty universal, which is partially by design, partially I think it just happened to keep going.

I don't know. It's hard to compare and contrast to companies building products. I do think YC is an enduring institution, that doesn't have a lot of threats coming its way, as long as it can continue to scale just the fundamental day-to-day things. A lot of that is software, right? I think relative to the average investment vehicle, YC has bazillion times more software powering all of this, smart people powering this, engineers powering this, which is very rare in the world of investing.

[0:53:09.4] JM: Do you think they're aggressive – well, maybe this was deliberate, maybe not. I mean, they've had three fantastic CEOs. Michael Seibel is CEO now, right?

[0:53:28.8] IS: Yeah.

[0:53:30.9] JM: They've gotten rid of the God-king mentality, right? Because they have moved even in really good times, they've moved through three different CEOs. You have scheme and risk in some of these other companies, like Facebook and Amazon maybe. We don't know. I mean, probably these are super enduring institutions as well, but you prove that it's an enduring institution very early on when you have multiple successful CEOs.

[0:54:06.5] IS: Yeah, that's a fair point. I don't know how to attribute that, but I definitely think about it in terms of Harvard. Who's the president of Harvard? I honestly don't know now, and it probably doesn't matter who it was two years ago and doesn't matter who it's going to be 30 years from now, right?

I do think all of the people that have been involved have been people who grew up in the ecosystem, right? Sam, Seibel. Jeff maybe didn't grow up in it, but he's been part of it for a very long time. There is something and it's intangible, but there is this ethos, this desire to do right by the startup ecosystem, the founders to do the right thing and to optimize for the success of the companies that I think is pretty fundamental to being an influential person in the YC ecosystem. Keeping that as the bar has helped, right? I think, I definitely look at my approach to my career and my social group and who I like to work with these days, how I approach this investing business today. A lot of it is influenced by Paul Graham's worldview. It's not to be discounted.

[0:55:35.8] JM: We're almost out of time. Do you have an extra 5 or 10 minutes?

[0:55:37.7] IS: Yeah.

[0:55:38.3] JM: Okay. I'd like to talk a little bit about the current fundraising environment, the VC environment and the investment landscape that you focus on. You obviously have a lot of experience and developer tools. I guess, what I could ask is what are the biggest sectors in developer tooling, the areas with the most acute problems that you find yourself very focused on these days?

[0:56:24.5] IS: In-developer tooling?

[0:56:25.4] JM: In-developer tooling.

[0:56:26.8] IS: Yeah. I think I've been looking for something that can be summarized this Parse for machine learning for a very long time. Machine learning has become an integral part to a lot of products, business's approaches to new things. I think it is very much the Wild West in terms of how to get the leverage of it, without all of the nitty-gritty headache. Putting aside the science, putting aside the math, the actual tuning of features and building models, just like the wrangling of the data, the hosting of the models, the training cycles, the whole stack up and down, I think is super messy and it doesn't have great best practices and everyone reinvents the wheel.

There's lots of projects where their actual models are not that sophisticated, or not that differentiated, but all of the infrastructure they have to build around them just bogs everyone down and it's just this monstrosity of maintenance and engineering.

I've been looking for interesting companies up and down that stack. I haven't actually found many. It's a challenge. A lot of the startups that I think enter that world and try to build an elegant tool, or framework, or hosted service, or whatever get bogged down with consulting and services. Because a lot of the people who engage with them aren't even really ready to consume something well-packaged and well-defined. They really just need help with what models should we build? What challenges can we solve with this? How good is it? People get bogged down in these services businesses, that one day hope to be the next great platform.

Anyway, that's an area I found really interesting. We were talking about this earlier, I think the world of data infrastructure more generically is really interesting. What happens now that we have these big cloud data warehouses, like Snowflake, Redshift, BigQuery that are really scalable, really cheap, reliable. What can you do with data once you centralized your data there? First, you have to centralize your data. One of the companies I've invested in is called Fivetran, which is really interesting. It has an elegant maybe Parse-like approach to that.

Then what can you do with that data, right? What kinds of BI tools can you build on top of that? What operational tooling can you build on top of that? I think this is interesting change in the world that hasn't really been internalized yet. Those are two of the areas that I think are interesting today.

[0:59:28.3] JM: The point you made about companies that often, they can end up feeling services businesses that are hoping for a time when their services business turns into a more of a product, or a self-serve thing. How do you differentiate between a company that is going to be in that state forever, like that's going to be in the service like, "Yeah, we're helping you integrate your data infrastructure." We've got this tool and we're hoping this tool turns into the self-service thing. How do you differentiate between the ones that are going to be in the services business for a long time, versus the ones that aren't? I guess more importantly, is that even relevant? Pivotal is still in the services business, but it's – I think, it's a pretty good business, right? I don't know. Does that factor into your investing theses?

[1:00:38.2] IS: Yeah. I mean, I think there are lots of services, heavy businesses that in the absolute, are really good businesses. Maybe if I were investing my own money, still I would want to be a part of. We're looking for venture scale businesses. I think to get the types of outcomes, the types of margins that I think are expected, it's tough to have a heavy services mix. It's tough to scale it, right? You're fundamentally bound by smart headcount. In this case, some of the most expensive headcount that exists on our ecosystem, right?

If you're a machine learning consultancy, how many people can you retain that aren't good enough to be hired by a product company that can get leverage off of machine learning? There's just all these inside of mismatches that I think are tough. In terms of how to think about it, I don't know. I mean, I think you really have to have judgment around what part of the stack the team is tackling, whether that's an important part of the stack and can be built off of, or whether it's one small piece in a long chain of stuff and they're going to get bogged down and integrating the other parts of the chain.

Talking to customers helps a lot, so sometimes you can't do this because it's a seed investment and they're just getting started. For the series A, typically there's at least half a dozen customers you can go talk to. It helps to understand the journey of how they have engaged with the company, how much of the value they're getting is from the product, the platform aspect of it, versus the expertise that is layered on top of it, how that's changed over time.

I don't know. I was diligencing a company recently and I thought it was pretty interesting, but I talked to some of their customers. At the end of the day, I heard from a few of them, I was like, “Oh, yeah. It was great.” They solved my problem for me. I learned about the value of machine learning from them. When I dug into it I’m like, “Okay, how's the product really working for you? What's the next step here?”

I heard things like, “Oh, yeah. It works pretty well,” but I think I've realized I can hire machine learning people myself and I'm going to be able to blend that with my expertise of my business domain that sits in-house. At the end of the day, all they were doing was stringing together open source anyway, and so I can string together open source in my world.

I mean, it doesn't come out that easily. When you peel back the onion on some of these things, that's the fundamental truth. Some people still make progress out of that and sometimes it's unavoidable. I think you have to be careful.

[1:03:44.6] JM: Last question. If you had to leave venture right now and start a company, what would you start?

[1:03:53.2] IS: Interesting. I try not to have that answer top of mind, because then it gets – sometimes – it's cognitive dissonance.

[1:04:11.1] JM: Well, especially because you're coding on the side. You're going to be like, "Oh, maybe I'll just check this thing."

[1:04:16.1] IS: Yeah.

[1:04:16.7] JM: Let's just integrate Strip and why not? Oh, no. It's taking off.

[1:04:19.9] IS: I think this is not super unique, but I find the resurgence of activity excitement funding around horizontal productivity and collaboration tools to be really cool. You see things like Notion, Airtable, Retool. I think these are these are category of companies that I think are very cool, something I get excited about. I think I have some ideas that I'll probably keep to myself, in terms of what can be built maybe in the gaps between those products, or taking those products as a premise.

I think that category of stuff is really cool to me and I like funding it. I've worked with a number of companies in that ecosystem. One is called Parabola, which is visual programming for everyday business people that are good at Excel, or Google Sheets, but not ever going to write a Python script. I think that's cool ecosystem. Part of me wants to maybe redo some of the Parse journey, but part of me is like, "Uh. It's in the past. Let it lie." Hope someone picks that idea up again and runs with it.

[1:05:34.3] JM: Okay. Ilya, thanks for coming on the show. It's been great talking.

[1:05:36.5] IS: Thank you.

[END]