EPISODE 08

[INTERVIEW]

[0:00:00.3] JM: Pete Punt, you were a long time engineer at Facebook. You founded Smyte. Smyte was acquired by Twitter. We're now here at Twitter. Welcome to Software Engineering Daily once again.

[0:00:08.6] PH: Hey, it's great to be back, Jeff.

[0:00:11.0] JM: When you joined Facebook, the engineering organization was quite small. It was around a thousand engineers. What was the engineering management structure when you joined Facebook?

[0:00:20.6] PH: Oh, man. I joined Facebook back in the beginning of 2011. It was actually right after the movie The Social Network came out. I remember I was in grad school and my mom watched the movie at around the same time I got the offer letter and she was like, “I don't know if this Mark Zuckerberg character is going to be a good person to work for.” It ended up being a great experience.

Just to get a sense for when in history that was, it’s a long time ago now. When I joined, I think the whole company was around 3,500 people, something like that. I think engineering was a little under a thousand in a separate building in Palo Alto. When I joined, I had an – like I recently converted eng lead turned into a manager. A person that probably preferred being a programmer more so than a manager, I don't actually know. That tends to be the archetype of that type of person. I think, maybe two levels down from the VP, something like that. That makes sense?

[0:01:21.6] JM: Yeah, it does. Was Facebook developing its engineering management structure at the time? Was it trying to emulate Google or Microsoft? Did it have a strategy in place, or was it just winging it?

[0:01:34.1] PH: Well, I was pretty junior at the time. I just kicked off my career. A lot of those decisions were opaque to me. They had just come off of this big code quality company objective. I arrived right at the end of modernizing the development practices and I think that there was somewhat of a rebuild of the management practices as well.

When I was there, boot camp was already in place. They had some management training programs already in place. It was actually – there was some maturity to it and I don't know if much has changed since then. Certainly, I was there for almost four years and that type of thing didn't change that much, just got bigger.

[0:02:18.0] JM: When you joined Facebook, you went through boot camp. In Facebook boot camp, you learn a lot about how Facebook operates as an organization. Was there anything you learned early on that surprised you?

[0:02:31.7] PH: Yeah. I mean, the things that Facebook did with vanilla PHP was pretty impressive. I mean, they built this whole massively distributed SSH client that would run across a 100,000 nodes in multiple data centers. That was written in – it was a single PHP file that copied itself to every server. It was totally crazy. I think that's the first thing that stuck out was how far they pushed PHP.

Other couple things that were a little surprising to me coming from grad school, which may or may not be depending on where you've worked, all the development was done and I think continues to be done on remote dev server. They would give you a shared big, beefy box in a data center, then eventually you would graduate to your own allocated box in the data center.

With a lot of the challenges with developer velocity, the first thing that they solved it with was just giving the machines lots of CPU and lots of RAM. That was another thing that somewhat surprised me when I joined. The move fast culture is very, very real. I mean, they take it super seriously. I had worked a full-time job before, or in-between undergrad and grad school and I sat around for a couple weeks waiting for permissions to get access to the repo and to get access to production. I mean, I wasn't productive for my first month, because I just didn't have permissions to do anything.

At Facebook, they have you landing commits on them on master in the first day and you're in production. If you're not in production in your second week, or queued to go to production in your second week, you get pulled aside by your boot camp mentor and they're like, “Hey, you got to speed it up.”

[0:04:14.5] JM: Out of boot camp, you started working on videos, Facebook videos. People may not remember, but back in 2011 video was not as pervasive on the internet and it was not as easy to work with. It was not pervasive and there certainly weren't easy tools for working with video, at least compared to today. What was hard about working with video in 2011?

[0:04:38.8] PH: There was a ton. If you can remember that far back, I don't know if streaming and Netflix was particularly popular. Hulu was just starting to get popular. It was in that era when you would watch Hulu and the same – I mean, this happens a little bit today, but the same ad would come up every single time because they couldn't fill the inventory. Video was like a hackathon project from two or three years before I joined and nobody was maintaining it. It wasn't a source of revenue for the company. They didn't even have that on the horizon.

It was a lot of fun, when you work on something that is important and millions of people use, but does not have a bunch of PMs on top of it, or a bunch of execs watching everything that you do. You get a lot of freedom to work on whatever you want to work on. That was a lot of fun. The big challenge when we joined was, or when I joined was I joined a team of one other person who transitioned off of that team to go work on mobile, just like a lot of other people did at that time, because there's a big pivot to mobile maybe six months after I joined.

The first challenge was every Sunday morning, I would get paged pretty much for this service. Again remember, this is Facebook, so most stuff is written in PHP and is in this model with a gap, which means that there's a team that handles a lot of the pages for you. A lot of times, your average product developer doesn't have to deal with being paged.

For this service, again it was a hackathon from a couple years ago. Somebody hacked out this Python script that ran on top of a single MySQL database with this ad hoc implementation of Amazon SQS, right? Something like that. The queue would get backed up every Sunday morning for some reason. The explanation we had was people were uploading their videos from Saturday and Friday night all on Sunday morning. I don't know if that was actually true.

Anyway, the queue would get backed up and then I would have to go and restart all the encode nodes just manually. One of the first projects that I did was basically try to modernize that codebase, which was take this custom Python MySQL-based solution and move it to the standard Facebook SQS type of thing, like task queue. That meant basically porting the whole thing from Python to PHP and running it in the monolith.

That ended up being like a net reduction in lines of code; what we call the async tier team, which was the SQS type of thing ended up dealing with a lot of the on-call issues. It made everything a lot better, because it was a real maintained service, not this ad hoc Python script, which by the way lasted us three years as the second largest video site in the world, which is cool.

[0:07:30.6] JM: In your work on video, you integrated the video product with timeline and the photos products. These were really high-visibility products. What was the process for getting your new features, your new products merged into products that were so important, because at many places this would be a multi-step, high-sensitivity process with lots of management layers that would have to sign off on things, so what was the process of Facebook?

[0:08:00.0] PH: Yeah, that's a great question. There were some things that were opaque to me, like where the designs came from. We would generally work with a designer who would go off and hang out with all the other designers and come back with a PSD that would say, “Hey, here's the product we're building. Let's go build it.” With respect to timeline, there were two things, two big initiatives that I remember back in 2011. Maybe you could say three.

The first one was we were getting ready for the IPO, and so there was a lot of work, battening down the hatches, so to speak. The second was timeline was about to be announced, so there was a huge amount of work into that. Then the third one was this big shift to mobile, which happened towards the end of 2011. Those are the three big initiatives. With timeline, there was a team that was sent to go build timeline and they had a PM, a designer and a whole set of engineers to go do it. Then once they built the V0 of timeline that shipped to employees, the photos and videos parts of that started being handed off to us and they were like, “Hey, now do the real version of this, so to speak.”

They had built a V0 photos integration and then we took it over and rebuilt it. Photos and videos were very, very tied together at Facebook. I very quickly started working on photos throughout that process.

[0:09:25.5] JM: Not long after you started your work on photos, you were a lead engineer, or you were leading the team to some degree?

[0:09:34.0] PH: Yeah. I mean, I thought it was doing a good job, but I got lucky, I guess. I was working on video, started helping out on photos and then got to know the photos codebase really well. Right around then, Zuck says, “Hey, smartphones are going to be a big deal. The vast majority of Facebook users are going to be on mobile really soon. If we miss this opportunity, we're going to have a bad time.”

You start pulling all these people off of the web and middle-tier teams to go work on mobile client and just through tons of people on it. My team turned over pretty quick. A lot of people went over to iOS or Android to start building the photos product over there, or build the newsfeed product over there. Very quickly, I was the most tenured person on that team after having been at the company for a year.

[0:10:28.3] JM: This is web photo specifically.

[0:10:30.3] PH: Yeah. This is web photos and the domain logic around photos. The mobile client will talk to some PHP endpoints and says like, “Hey, give me all the photos.” Then there's a bunch of privacy checking and figuring out who's tagged in what. We owned the code that generated that HTML, CSS, JavaScript, as well as the data fetching and some of the privacy code.

[0:10:52.1] JM: Okay. All the mobile teams had to interface with you also.

[0:10:53.9] PH: Yeah, yeah. That's right.

[0:10:55.1] JM: Okay. Was there something that you had done in terms of yours your strategy, or your diplomacy, or just showing that you could write code really quickly and interface with people productively that stands out as – because you joined the company around 2011 and then by the end of 2012, I guess you were leading this photos team, or was it just a matter of attrition?

[0:11:20.5] PH: I mean, attrition had something to do with it. I think I stepped up to the plate for sure. I think there's a couple of things; being available and responsible, or responsive to customer requests is huge.

[0:11:31.7] JM: Even on Sunday.

[0:11:32.9] PH: Yeah, even on Sunday. That's right. It was it was the type of thing where a mobile engineer has a question about how we do some rendering of photos on web, because they're trying to rebuild it on mobile. I would be in IRC at the time, or whatever it was we were using the Facebook groups later on. They would ask the question and I would answer the question really quickly.

Very quickly, people start associating my name, or whoever is answering the questions with the product. Then you start to understand the customer needs really well. As long as you're still actually pushing the product forward at the same time that you're talking to customers, that usually gives you a really good perspective on what people want and what the rough edges are and the API, or the service that you're maintaining. I think that was a big part of it.

I also had a really good manager who told me, "Hey, Pete. You've got a lot of potential, but the manager can't bestow upon you the respect of the team, or bestow upon you the responsibility of promotion, unless you're actually already at that level." That was a pretty good piece of advice that she gave me.

[0:12:40.5] JM: One thing I've noticed talking to other Facebook engineers is there is this notion where engineers become engineering leadership, but that does not preclude them from having managers. When you have these engineers that become leaders to some degree, but they still have managers, what's the dynamic there? What's the role of the manager and what's the role of that engineer that has established clout in the company?

[0:13:08.8] PH: Yeah. I think the relationship between an engineer and a manager is a really interesting one to talk about. I've noticed that it's different at different places. For example, I'm at Twitter right now and I think Twitter has a lot of – the way that Twitter treats engineering managers is similar to a lot of companies, which is engineering managers are responsible for the backlog, oftentimes they're that product owner, so engineering managers at Twitter, we have final say on what gets in and what doesn't get into a particular product.

At Facebook, it wasn't like that at all. I think different teams at Facebook operate in different ways, so I can't speak for every team. In my experience, the engineering managers were pretty much just coaches and they didn't really have too much of a say in the actual work that was being done. We would work directly with a PM and the PM would run the weekly, or twice a week meeting that says, "Hey, here's all the tasks we need to do. Here's the prioritization. Let's go through it." A lot of it was self-directed as well.

When Facebook works well, Facebook the engineering organization works well, the PM comes in and says, "Hey, here's the objective we want to meet." Then the engineers just go and figure out what they need to do, get approvals on the thing that they're going to ship if they need to and then they ship it. In that way, the engineering manager's responsibility is mostly recruiting, performance management. If you're doing a great job, we make sure that you're recognized for that. If you're not doing a great job, we work with you to try to get you to be doing a great job.

Unblocking various types of coaching. If you aren't making progress and you're always blocked by another team, putting the right people in the same room to have that conversation. Generally, a pretty hands-off management culture. In terms of the responsibility of senior engineers, yeah, I mean, that's a pretty broad question. I don't know if there's something in particular that you wanted to focus in on.

[0:15:14.1] JM: Well, I'm just wondering because there are places where the – in fact, most places the manager is in a role where the engineers are deferring very much to the manager and that's how it would sound. The manager manages the engineer, the manager leads the engineers. My sense is that at least in some early days of Facebook, the product was just – had so much traction. Everybody was moving really quickly and I guess, there was an extent to which the engineers were leading the development as much as the managers were organizing the engineers.

[0:15:51.9] PH: Oh, yeah. I definitely think so. I think there were cultural aspects to that. I think number one, so when you look at the structure of Facebook engineering in those days, it looks a little chaotic. There weren't much in the ways of okay ours, and if there were, nobody knew what they were. There was a culture of moving really fast. Everybody was committing on the same codebase and there was this strong sense of shared ownership of the codebase. You would never get into a situation where you own the photos codebase and I own the videos codebase and I need to make a change that puts videos in albums. This was a task that we did.

We never got in a situation where the photos team would say, “No, you're not allowed to commit to my codebase.” Oftentimes, they would be added as reviewers, but the cultural norm was anyone can land code in any part of the codebase and that's fine. When you combine a clear sense of direction, so Facebook the product had a very clear and simple direction that it was going. With this culture of moving fast and shared ownership, that leads to a lot of technical decisions being made at the leaves of the tree. Does that make sense?

[0:17:05.5] JM: Yeah. You joined the Instagram team shortly after the acquisition in 2012. Most tech acquisitions fail. The Instagram acquisition was perhaps the most successful tech acquisition in history.

[0:17:20.5] PH: YouTube was pretty good.

[0:17:21.8] JM: YouTube was pretty good, okay. We'll see. I guess, this is something we'll have to check back in on in 10 years. Tell me the story of the Instagram acquisition from your perspective.

[0:17:32.0] PH: Yeah. If you rewind to end of 2011, do you remember the app Facebook camera?

[0:17:42.2] JM: No.

[0:17:43.0] PH: Okay. Nobody does. We had this app, Facebook camera. I remember, this was – I started to develop some strong opinions around this project. Facebook was always about ship the minimum viable product. If you're not sure whether you should ship or not, you should ship, get the data. Ship it to a small test group. Get some data, roll it back if it sucks. The whole company was organized around that.

When we started the shift to mobile, the shipping velocity really went down a lot and Facebook camera was a great example. Instagram was getting really popular. We knew there was a big opportunity in mobile photos. We were building a bespoke photo-specific app that was called Facebook camera. I remember, just this thing just never shipped. It was just taking a really long time to ship. I remember, sitting in some of those presentations where they'd be like, “Yeah, we're trying to decide whether you should swipe to reveal the camera, or press a button to reveal the camera.” I'm just like, “Just ship one of those things and we'll figure it out later.”

Now there's a school of thought, which is the Apple school thought, which is ship a very polished, very beautiful, very perfect mobile app. A lot of companies have been really successful doing that. I was like, a year or two in my career. I don't claim to be right or wrong about that, but I think it was something to consider.

Anyway, so we ended up shipping Facebook camera. Then maybe three months later at 8 a.m., our VP engineering reaches out to our team and I was on the server side part of Facebook camera and they were like, “Hey, we got a big announcement. Just so you know, we're going to announce in an hour that were acquiring Instagram and they're going to come to the office later today, so please give them a warm welcome.”

This was a company that we were viciously competing with. Facebook as everybody knows is a very proud company. When Facebook was – everyone in Instagram was kicking their ass, like everybody was taking it personally. It's like, “Oh, these people are now your friends.” It's a little weird. They showed up and we got a lunch, or coffee with them. I remember, there's a photo that a bunch of – a lot of the people on the Instagram team are now my friends. There's a photo that somebody took from the balcony of them walking in clearly bewildered, because I think a lot of the employees didn't know they were getting acquired until that day too.

Everyone's like, “Oh, what's going on?” That was a weird day. It did end up being a really successful acquisition. I think, I learned a lot watching that acquisition happen and being somewhat a part of it as I was the first individual engineer to go over full-time to work on that team. It was really interesting to be part of that. The first thing that I think made it work was they put Instagram –

We had these garage conference rooms in the campus, which were really nice. They were trendy garages that were totally nice. We just put them in that garage space. It was basically like, “Hey, keep doing what you're doing. We're not going to go in and make a bunch of changes immediately.” I don't know what types of conversations Mike and Kevin were having at the time, but for your rank-and-file, nothing had changed, except they got nicer laptops and HR department, that kind of thing.

Then I think the pitch to them was, “Hey, we're just going to give you a ton of Facebook resources to go make your product even more successful.” I was one of those resources. By the way, I hate calling people resources. It's one of my big pet peeves. I think resources in that context is data centers as well and reach and an audience and stuff like that.

I went over. My role was to build out a lot of the web stuff. I've been working on – Facebook likes to, or at least back then, liked people to rotate off of teams every 18 months, 12 to 18 months. I was hitting the 18-month mark I think, or maybe a little bit beyond that. They were like, “Hey, how about you go try out? Try being the first person to go over to Instagram?” She was a lot of fun.

[0:21:54.9] JM: At Instagram, you were working with AWS infrastructure, whereas most of the Facebook infrastructure to that point was homegrown. What was the difference between building on Facebook infrastructure versus public cloud?

[0:22:08.6] PH: Oh, it was totally different. I came in and I was pretty surprised at what they were doing. There were some things that I thought Facebook was doing better, right? I mean, Facebook was a more mature company. For example, better developer tooling around tests and CI, right? Facebook had this really great data fetching and caching and privacy layer that Instagram was a much simpler app and it didn't need. Its privacy model was way, way simpler. It wasn't there and it seemed like something that I would have expected to be there.

There were some things that were really great. Facebook, or sorry, Instagram was on continuous deployment, or at least something close to it. Any engineer at any time could run this command called Yolout, which would push the site out. I thought that was crazy, because the way that Facebook did deployments back then was every Tuesday at 2:00 p.m., every engineer would have to say, “Hey, I'm in,” in an IRC room. Then sit around for three or four hours as the site was gradually pushed out to everybody. If you weren't in IRC, we just revert all of your commits and roll out that version of the site without you.

The fact that Instagram was on continuous deployment was shocking to me. I was like, “These startup people are crazy, but ended up being awesome.” I think Facebook is now on continuous deployment as well.

[0:23:40.2] JM: What did you learn about management when you were at Instagram?

[0:23:44.6] PH: Well, I was an individual contributor there for over a year, before I switched over to management. I had been taking a lot of the management, the Facebook management courses and stuff like that. I knew at a high level how it worked. Again, we took the – I managed my team at Instagram the way that most teams at Facebook were managed, which was the PM is the product owner, the designer co-owns the product with the PM. The engineering team goes and executes. Your role as a manager is to step back, let the process happen and only insert self into the process when something seems broken. That's how I ran the team there.

You've got to just do management for a while to really learn it. How to run a one-on-one, how to spot problems before they escalate? Somebody tells you something, how to get to the bottom of where that comes from. That was all stuff that I learned on the job at Instagram.

[0:24:51.4] JM: Well, is there anything that had been opaque about the management structure beforehand that was revealed to you during that management training and learning the ropes of being a manager that clarified how management had been working at Facebook?

[0:25:11.1] PH: Yeah. Let's see. Number one, I think all of the best managers I've ever had are professional managers. As in, they don't write code anymore, they don't want to write code anymore. They're still technically sharp and you can't tell them something that's totally wrong and they – they'll understand what you're doing, but they have no desire to write code or weigh in on technical decisions. Those are the best managers, because they're focused on the stuff that you, the individual engineer aren't focused on.
How do I put together a good case to be promoted? What skills am I lacking? A lot of engineers are really good at writing code, but the instant you ask them, "Hey, why did you do that?" Even if they know that they have a good reason for doing it, a lot of times it's just hard to articulate that. I think a good manager says, "Hey, here's how you would articulate this. Let's talk about how you would come to the –" It's mostly a communication and a psychological safety job, I'd say. That was one thing that I learned. Value and focus on recruiting is another one. I think recruiting super important.

[0:26:24.4] JM: React was open sourced around this time. Why was it valuable for Facebook to start open sourcing its projects?

[0:26:32.7] PH: A lot of people thought it was. Facebook had done some open source work in the past and threw some stuff over the wall and let it rot. The open source community actually hated Facebook. I mean, if you go Google for the 320 iOS framework and how Facebook treated that project, that's the context for the animosity and the developer community towards Facebook at the time.

When I went over to Instagram, they wanted – they said, "Hey, we've got all these photo pages. We got photo pages, profile pages, news feeds that kind of thing, written in mustache templates and jQuery rendered in this Django app that runs on AWS." For a number of reasons, the infrastructure team told me that it's got to be client-rendered and it can't be server-rendered. They said, "Hey, just go deliver these. Go finish these products and deliver them." I went to our UIE team, User Interface Engineering team that was Tom Okino is managing it, Jordan Walk was on the team and I asked them, I said, “Hey, we got to client-render something. I haven't seen us use jQuery at, or mustache at Facebook. What's the state of the art for client rendering?”

They said, “Well, we've got these three or four different projects and the incubator that we're working on. We've got boltJS, JSHTML, there's some other thing,” because we had bought the web OS team or something and they came with some framework. That framework turned into boltJS. We've got this thing, React, which we used for one tiny little experiment. I went and I diid an eval and the React getting started doc was super impressive. I mean, it's magical when you first use the thing. You're like, “How does this thing possibly work?”

I made the call to go with React. First thing we had to do was do an internal open source from this PHP-centric world that was diverged from the open source mainstream and pull that over into the Instagram stack, which was super circa 2012 startup, open source mainstream. That was a lot of work. Once we got the initial web app launched, we were like, “Wow.” I mean, there's a ton of rough edges, but this was way better than the existing jquery, mustache thing that came before it. There was just a lot of appetite on the Instagram side, as well as the Facebook side to get this thing open sourced.

[0:29:04.7] JM: With GraphQL, I don't know if this was the case with React. With GraphQL, when I talked to Lee and Nick, they told me that they took a long time to make sure that the open source announcement would be successful. It was in contrast to the move fast and break things approach. There was a protracted release process for GraphQL. Was that the case with React also?

[0:29:30.7] PH: Well, getting it launched and all the internal stuff that you have to do took a little while. You got to talk to legal and you got to package it up and you got to build a website and documentation. There were many nights that me and Christopher Chedeau and I think, Tim Young was there as well. We're just putting in the work. With respect to the launch plan, it was basically, I think we had sponsored JSConf and Tom Okino had a keynote slot that he could allocate. We're like, “Yeah, we'll launch it JSConf and everybody will think it's great.” That's what we did.

Again, the developer community did not like Facebook. The response was not good. It was very much negative on React and it was this whole thing that we had worked so hard on, we were all really excited about launching this thing and the developer community was like, “Facebook's a bunch of clowns. You guys are using XML. Even worse, you're putting XML on the JavaScript. What is going on? You’re not separating concerns.”

I don't know. I don't want to put words in Lee or Nick's mouth. My guess is they probably remember that launch and they were like, “We're not going to make the same mistake that React made when they launched.” Now we were able to recover it six months later, but it was still – it wasn’t the most fun to see on Twitter; the response to that announcement.

[0:31:00.6] JM: React has developed a really strong following online. It superseded Angular as the most popular framework from my point of view. Was React actually superior to Angular, or was it just marketed more effectively?

[0:31:21.2] PH: Oh, man. You’re asking me to go on record about this? I think it is.

[0:31:27.1] JM: I would argue that those things could be one in the same. If you have good developer documentation and you have good outreach strategies, that is equivalent to marketing and that is what wins people over.

[0:31:37.3] PH: Yeah. Yeah, that's totally –

[0:31:39.1] JM: There's network effects.

[0:31:40.3] PH: Yeah. I agree with that. I think, you're being irresponsible if you're open sourcing something and trying to convince people to use it. If you don't truly believe that, it's better than the solutions that are out there. Yes, everybody working on React thought that we had a better mousetrap, at least for a decent subset of the population than what was already existing. I'll caveat that with I think that if you look at the popularity of Angular and then the rapid rise of Vue.js, there is clearly a segment of the web development population that wants to write markup, annotated with some magic attributes to bring their webpage to life.

That is not what I want to do. That is not what most people that come from a CS, or software engineering, or even hobbyist programmer background want to do. For those that were – that cut their teeth on HTML and CSS and that's where they feel most comfortable, there's a market for that. I think for people that have the title software engineer, or think of themselves as programmers first, not web designers first, React was clearly a better technology across a number of different dimensions.

[0:33:02.5] JM: When you left Facebook, you started Smyte. What adjustments did you have to make to how you thought about building product and building teams when you went from a large organization like Facebook to building your own company?

[0:33:17.2] PH: Yeah. The first thing that you notice when you leave to start your own company, and this was especially true for me, because I had done a lot of advocacy and development on React and I had a bit of a personal brand. I was like, "Oh, everybody's going to want to work for me, because I'm so great." Wrong. Hiring is actually super hard as a startup, no matter who you are.

The Facebook engineering brand and boot camp and all the great sourcers and recruiters and closers that they had there, which we just didn't have, you take that for granted. When you leave, that's the first thing that you notice. The other thing which might be more Smyte specific is I went from a very large-scale B2C application to a to a small B2B startup. The way that you interface and get value from customers is different.

At Facebook, we would launch a product, we would watch the graph go up and get really excited and then we would watch TechCrunch write about our new product launch. At Smyte, we would actually get to talk to the customer and we would get to know them a little bit and we would get really great, high-quality feedback as opposed to at Facebook where you look at all the things you're measuring and you try to guess what people actually want. We could actually talk to the customer, which was nice. Also, we would never get write-ups and in publications or anything like that. It's just different. I think the big thing is recruiting.

[0:34:56.3] JM: Eventually, you got acquired by Twitter. We talked about the Smyte company experience in a previous episode to some degree. Now that you're at Twitter, you get to see how the product development process differs between arguably the two leading social networking companies. How does the product development process at Twitter contrast with that of Facebook?

[0:35:20.2] PH: Yeah. The first one that I feel – I manage a couple engineering teams. The role of an eng manager here is much different than at Facebook, at least in my part of the world. Again, I'm expected to be the product owner at Twitter. At Facebook, the PM and the designer were more the product owner. That's just a different feeling. From a technical perspective, Twitter went all-in on microservices. If you don't know the story of Twitter, basically there was this big monolithic rails app that would fall over all the time. The company sat down and said, “Hey, we got to go fix all the fail whales,” which is what they would call the error page.
The solution is we're going to break the monolith into multiple independent services. Because we were going to be rewriting the whole site anyway, let's rewrite it in Scala and run it on the JVM. That is the current state of things today, is that every team runs its own couple of services generally written in Scala. All the services talk to each other, contrast and there's strong ownership on those services and code bases.

Contrast that with Facebook and remember how I was talking about how – there's an expectation that anybody can land code in any part of the codebase and you're supposed to accept the diffs. That's very different. Facebook was largely, at least to the product engineering team, which I think was basically the biggest engineering team, hundreds of people all on one big monolith the app, all committing code where they need to commit the code to deliver the application. That I think from a technical level is probably the biggest difference.

[0:37:12.7] JM: What's so amazing about the Facebook story is the degree to which the Facebook engineering team managed to bend PHP to its will, or at least that's one of the things that impresses me the most and like, I've talked to Keith Adams a couple times about the HHVM stuff. That was a unconventional approach from my perspective, like what Facebook ended up doing with PHP.

Here you're saying that perhaps, there was some detriment to breaking up the Twitter monolith into microservices. It's interesting, because the Facebook story presents an unconventional decision that worked out well. The Twitter story presents perhaps a conventional decision that worked out less well. Has it given you any insight into how to make engineering decisions according to “best practices”?

[0:38:10.9] JM: Well, I mean, the title of my first talk ever was rethinking best practices. You've got to have a respect and understanding for what the best practices are, and then know when to bend those or break those. The way that I think about this is let's think about why, and there's so many interesting topics to talk about here. If you talk to Keith Adams, he says that PHP got the most important – like one of the most important things about PHP is its process model.

PHP gets everything wrong, except when you start up a new web request, you get a totally clean slate with no shared state. I wasn't around for the big monolithic rails app, but I think a lot of people's concerns with rails is that there's shared state between different people's requests and it's pretty hard to guarantee isolation between those requests, both in terms of privacy and resource utilization.

PHP actually had some positive characteristics. I think, look at why people split things out into separate services and take the micro-service approach. It usually comes down to our organization is structured this way and we can move faster if we own our one piece of the codebase ourselves, we can deploy whenever we want, we can talk to other teams through these stable interfaces, so it gives us a lot of freedom to go and do our own thing without being blocked by other teams. Is that a fair representation of the micro-service argument?

[0:39:50.8] JM: Yeah. I would say so. Yeah.

[0:39:52.2] PH: Okay. I think that's great if you know what your org structure is going to be over some time horizon, or you know what the product requirements are going to be over some time horizon. The problem is as anybody that's working in a big company for more than a year knows, reorgs happen all the time, product direction changes all the time, and so you end up with these services that you have to maintain forever that match the org structure and the product requirements from five years ago. Today, they don't make a lot of sense.

I don't know whether it was intentional or not, but the Facebook approach basically optimizes for future unknowns. It embraces hey, we don't know how to chop up the application in a way that will work for tomorrow. We're going to just assume that we're going to have to coordinate across different systems in different parts of the codebase. We're just going to make that coordination as cheap as possible, as opposed to the micro-service approach, which is we're going to solve the coordination problem by just not coordinating, or by minimizing the amount of times that we have to coordinate.

I think that in my opinion, embracing the future unknowns and just saying, “Hey, we're going to have to coordinate a lot. We're going to minimize that coordination cost,” is the better approach. You see this with a lot of people, or a lot of companies that have moved from lots of small git repos to one big mono repo. I mean, I know that this is still debated endlessly. At the end of the day, people don't like to version code. If we have all these different git repos, then we've got to be very careful about when we push a new release, what features – what are the breaking changes, that kind of thing. That's an example of there is coordination between even these separate services that has a cost and you got to minimize that cost. Is that making sense?

[0:41:45.5] JM: Yeah, absolutely. Taking a step back, you're a connoisseur of social networks. You've used all of them. You've been a part of many of them, most of them, well, the biggest ones by volume.

[0:41:58.9] PH: I have friends at Snap. That’s the only one –

[0:42:00.2] JM: You have friends at Snap, okay. Tell me something that you believe about social networks that I would not hear anywhere else.

[0:42:10.4] PH: Oh, boy. That's a hard one. I mean, some of the use cases for these networks are really surprising. As somebody that's worked on this stuff for a really long time, I think social networks are in a lot of ways a mirror and an accelerant. A lot of the concerns that people have about social networks are about people. They're fundamental human problems that we've been struggling with for a long time.

The only difference between social networks and the way that things – we've evolved as a society before, or that social networks just make everything faster. That's my perspective on – I don't know if that's different. I mean, I think probably many people have said that before, but that's a hard question.

[0:43:07.5] JM: Fair enough. We have entered this time where we've moved beyond the era where social networking was zero-sum. Facebook arguably took MySpace's launch, tumblr has lost some market share. Most of the other social networks seem to be growing in a nonzero-sum way. LinkedIn feels like it's growing. Quora feels like it's growing. Twitter, I don't know if it's growing by usage, but user interactions, but it certainly feels it's growing in terms of its prominence and its usefulness.

[0:43:41.4] PH: Yeah. It's actually – we're doing great. I'm really excited about being here.

[0:43:45.4] JM: I believe it. I mean, I'm not –

[0:43:46.9] PH: It's one of the situations when I love the product and then I came inside to see how the sausage was made and I was like, "Oh, this is a lot better than expected."

[0:43:56.0] JM: Twitter's the only app I've had to uninstall from my phone to restrain myself. I'm typically a disciplined person, but I've had to uninstall it. There's something fundamentally changed where we've gone from this world of zero-sum social networking to where we have all of these different social networking apps and it feels they're all serving different utilities?

[0:44:14.0] PH: Yeah. I think, so Twitter has positioned itself as – so if you go and install Twitter in the App Store, it's under the news section. It's not under the social networking section. Twitter positions as serving the public conversation, which is very different use case than Facebook, for example. Twitter doesn't want to be in the – or Instagram. Twitter doesn't want to be in the business of connecting you with your friends and sharing important moments with your family. That's a place for Facebook. Increasingly, it's a place for Instagram.

For Twitter, it's about talking about serious stuff and topical things. I use Twitter to connect with my industry peers and I use it to talk a lot about React back when I was really involved in the

React community. People use it to talk about public events. Have you seen the fire festival documentary?

[0:45:06.1] JM: Of course.

[0:45:07.3] PH: Yeah, there's a really good ad for Twitter at the end, where they're like, fire festival was marketed on Instagram, but the story about how bad it was broke on Twitter. The company got really excited about that one when we saw that in the Netflix documentary.

[0:45:23.6] JM: That's almost an understatement. It's a microcosmic understatement of how much impact Twitter is having. Is there anything – coming back to Facebook, is there anything from your time at Facebook and now that you've been outside of Facebook for a while and you have a deep understanding of what's going on across the industry, what is it about Facebook that other engineering organizations should emulate?

[0:45:49.2] PH: Yeah, that's a great question. I think that really emphasizing speed is important. A lot of engineering orgs, there's a big checklist of stuff you have to do. There's never really an item on the checklist that says, “Ship quickly.” I think, emphasizing the speed is important.

There's another thing that Facebook does. A lot of companies are basically modeled after Google, right? They take their promo rubrics from Google, they take their hiring and interviewing strategy from Google. One of the things that Google really values is doing technically challenging work and doing things that haven't been done before with computers. Facebook does not care about that. Or at least back when I was there, Facebook was all about customer focus, get as many people using the service as you can, bring value to them. If you can do it with a big ball of PHP and not doing anything clever, that's the best solution.

Eventually, when you get to half a billion monthly active users, you need to build a new type of graph data store. Then Facebook would grudgingly do that. As opposed to my stereotype of Google, which is everybody is itching to do that. I think that's an important cultural thing as well. Really optimizing – thinking about your engineering org and optimizing around what are the future unknowns with respect to coordination between teams and how do we hedge against that? Mono repo monolithic anything.

[0:47:35.3] JM: Critics who are reading this, or listening to it are going to be saying Facebook move too fast though and there were consequences. What do you think about the downsides of moving too fast?

[0:47:48.2] PH: Yeah. This is a conversation about software engineering and technical stuff. If you look at the criticisms of Facebook, they're not really around – I don't know the last time anybody – it's mostly around policy, right? Even in a lot of the most popular news articles about Facebook over the last year or two and it's been really, really critical, they've all been around hey, your platform policy was wrong, or so and so. Like use the API in a way that it was – the API worked as designed, but they then used the data in a way that was against policy and then you didn't go and go enforce your policy. I don't know. I'm not super close to the specifics there and I don't know anything that anybody else knows – doesn't know.

The fact is you want your leadership to make decisions and the engineering's job is to go and execute those decisions quickly and at high-quality. Facebook's engineering org is really, really good at that, in my opinion. There's plenty of criticisms about social media's role in society, plenty of criticisms about Facebook in particular and the decisions that were made. This is not my place to really talk about that. In terms of executing on that stuff really, really fast and efficiently, I think that everybody should strive to have that level of efficiency and execution.

[0:50:28.5] JM: Okay. Pete Hunt, thank you for coming on the show.

[0:50:30.4] PH: Thanks.

[END]