EPISODE 873

[INTRODUCTION]

[0:00:00.3] JM: Facebook moves fast because of vision, collaboration and trust. The fast pace of development is enabled by constantly improving infrastructure and a sense of unity throughout the company. In Facebook's early days, there was an emphasis on rapidly deploying new code to drive constant improvement and experimentation. Product quality was maintained by engineers closely checking each other's code reviews, rather than writing detailed unit test suites. Facebook engineers had a sense for how the product should operate and they were able to evaluate whether a new feature was working properly by testing a live version of that feature.

At Facebook, the vision of the company is clearly communicated to the employees. Every employee within Facebook can articulate the vision for the company and they will use similar language in describing that vision. Since the employees are aligned on strategy, they can also align in their implementation of product features. This reduces conflicts across roles and between teams.

Facebook has also shown a willingness to trust its engineers. Trust was exemplified by Facebook's tolerance for failures in the early days. When an engineer broke a build, or shipped a feature that failed to gain traction, that engineer was usually not punished. They may have even been rewarded, if the company could learn significantly from such an error.

Raylene Yung was an engineer at Facebook from 2009 to 2015. As she moved from individual contributor to manager to engineering director, Raylene worked on products including newsfeed, timeline, privacy and sharing. Raylene joins the show to give her reflections on the Facebook product and engineering environment.

She explained how Facebook's culture of collaboration, vision and trust drive fast product development and minimize conflict. Raylene left Facebook and joined Stripe, where she worked on payment systems and international expansion for almost four years.

© 2019 Software Engineering Daily

The new Software Engineering Daily app is live in the app stores for iOS and Android. It includes all 1,000 of our old episodes, as well as social features. You can comment on episodes and have discussions with other members of the community, including me. I'll be commenting on each episode to give some meta thoughts. If you want to have a discussion around these different episodes, you can jump onto the app, or onto softwaredaily.com to share your thoughts and you can become a paid subscriber to get ad-free episodes by going to softwareengineeringdaily.com/subscribe.

Also, FindCollabs is the company I'm working on. FindCollabs is a place to find collaborators and build projects. FindsCollabs is having an online hackathon with $2,500 in prizes. If you're working on a project and you're looking for collaborators, or if you want to work on a music project, or if you're looking for programmers or co-founders, FindCollabs is a great place to find other collaborators, or to post your projects and post your progress.

With that said, let's get on to today's show.

[SPONSOR MESSAGE]

[0:03:23.1] JM: You probably do not enjoy searching for a job. Engineers don't like sacrificing their time to do phone screens and we don't like doing whiteboard problems and working on tedious take-home projects. Everyone knows the software hiring process is not perfect, but what's the alternative? Triplebyte is the alternative. Triplebyte is a platform for finding a great software job faster.

Triplebyte works with 400-plus tech companies, including Dropbox, Adobe, Coursera and Cruise Automation. Triplebyte improves the hiring process by saving you time and fast-tracking you to final interviews. At triplebyte.com/sedaily, you can start your process by taking a quiz. After the quiz, you get interviewed by Triplebyte if you pass that quiz. If you pass that interview, you make it straight to multiple on-site interviews. If you take a job, you get an additional $1,000 signing bonus from Triplebyte, because you used the link triplebyte.com/sedaily.

That $1,000 is nice, but you might be making much more, since those multiple on-site interviews would put you in a great position to potentially get multiple offers. Then you could figure out what your salary actually should be.

Triplebyte does not look at candidates' backgrounds, like resumes and where they've worked and where they went to school. Triplebyte only cares about whether someone can code. I'm a huge fan of that aspect of their model. This means that they work with lots of people from non-traditional and unusual backgrounds.

To get started, just go to triplebyte.com/sedaily and take a quiz to get started. There's very little risk and you might find yourself in a great position getting multiple on-site interviews from just one quiz and a Triplebyte interview. Go to triplebyte.com/sedaily to try it out. Thank you to Triplebyte.

[INTERVIEW]

[0:05:42.8] JM: Raylene Yung, you are a former Engineering Director at Facebook, you're now at the Business Lead and Head of Engineering for APAC at Stripe. Welcome to Software Engineering Daily.

[0:05:52.4] RY: Thank you. Thanks for having me.

[0:05:54.8] JM: When you joined Facebook, you were working on newsfeed. From the other conversations I've had with engineers who have worked at Facebook, there is a saying at least that applies to the time when you were there, which is so the newsfeed goes, the rest of the Facebook product goes. My sense is that the newsfeed had a huge impact on the direction of Facebook. Describe how the product development work within newsfeed affected the rest of the Facebook product.

[0:06:25.3] RY: It's actually funny, I've never heard that saying. I think part of that must be when you're working on a product, your team, you maybe don't realize how people talk about it on the other teams. I would say the newsfeed was really an internal platform for many other Facebook products. This meant if you were working on something related to user actions, or

different types of content, you would have to interact with the feed as a system in some way. I think that was really – it was really quite interconnected.

[0:07:00.3] JM: When you joined the company, Facebook was starting the shift to mobile and this has been a pivotal point in some of the other conversations we've had. How did the shift to mobile impact your work on newsfeed?

[0:07:13.7] RY: I think it really affected everyone at the company at the time. I don't know that there is necessarily something very newsfeed specific about it. It's funny to think about this now, but at the time, I think there are very few engineers at Facebook and maybe even in the industry at large that had a lot of mobile development experience. There was the scramble across the company to figure out how to train and hire enough people to build everything that we needed to.

On newsfeed, perhaps that problem is even more complex, given there were many different front-end products on each of the different mobile platforms. Of course, we need to think about how they interacted with the existing backend data model and the web and mobile products.

[0:07:56.7] JM: Today, there are many products that have a newsfeed across the internet. What's the key to developing a good newsfeed?

[0:08:06.0] RY: It's a really hard question. I don't know that there's really one and easy secret, or key creating a great newsfeed. I think probably what's most interesting about the format itself is how varied it really is and how much it depends on the specific type of content that's part of your product, or even the audience.

For example, I think if you have a more visual feed, like one that's filled with photos and videos, freshness is a big factor, typically because if you've seen a video once, you probably don't need to watch it again. If your content is more discussion-based, or it's a comments – when new comments get added, that actually may warrant that content staying fresh and being on top of the feed. If you think of something like Reddit or Hacker News, it looks very different from YouTube. I think a lot of the key there is just keeping in mind the different types of content, what keeps it interesting and what your audience is looking for.

[0:08:58.7] JM: Do you recall any particularly hard engineering problems that you've worked on while you were on the newsfeed team?

[0:09:05.4] RY: Yeah, there are definitely a lot of things that are challenging. I think one of the earliest technical redesign projects I worked on was when we figured out how to aggregate actions and content when generating newsfeed. Just explained, I think we think of today newsfeeds as aggregations of content. At the time, every individual action or story that was created would be evaluated distinctly.

One example is, let's say Jeff commented on Mary's photo and that happened an hour ago. Then 30 minutes later, Bob comments on the same photo. Before these changes, newsfeed would look at each of those actions distinctly and figure out what to do with them. The project I worked on was to really redesign the whole product from backend to frontend that would already know how to combine those actions together. It would actually look at a joint blog that said, “Jeff and Bob commented on Mary's photo,” rather than looking at each of the actions separately.

I think, what was really tricky about this was it was very much a full stack redesign and you had to think both about what the UI and UX would look like, but also about how those data model changes would be represented in the backend and managed across simultaneous versions of the backend that was running. It also was a really long project, so I think that made it pretty memorable for me too.

[0:10:30.9] JM: What I find interesting about newsfeeds across the internet, I grew up using the internet and my ways of consuming information pre-newsfeed were so different. A lot of it was around forums and just going to, I guess Wikipedia-like sites. Post-newsfeed, it feels like I'm able to consume much more information about goings-on around the world and around communities. What's your perspective for how newsfeeds have changed how we consume information?

[0:11:09.4] RY: I think something that's pretty noticeable from what you described is the feeds of your childhood is the use of multimedia. I think today, when you imagine the newsfeed part, you actually see a pretty vibrant product.

You think of photo, you think of videos, you think of conversations sometimes happening in real-time. I think with feeds that focus on bringing a combination of that media and also keeping it fresh, thinking about how to show you the most immediately relevant content, I think it creates a much more dynamic view into what's going on at the time.

[0:11:41.1] JM: Was there a use of "machine learning" when you were on the newsfeed team, or did that come later?

[0:11:49.6] RY: The earliest versions of finding the right content for feed was pretty heuristic-based actually. A lot of it was around looking at the types of content and yeah. It was actually not that sophisticated at the time. Came later.

[0:12:05.2] JM: Okay. You started at Facebook pretty early in your career and you went from being an individual contributor, an engineer to becoming an engineering director. Describe the promotion path that you took at Facebook.

[0:12:23.8] RY: Yeah, it's hard to speak about this too generically, given I was at the company at a very particular stage of growth. Even as I was learning, taking on more, the ladders and levels can change year over year and evolve. Rather than go level-by-level, there are a few main milestones or things that happened. The first one was making the switch from an individual contributor role to a management role. I think that was a pretty gradual change at the time when it first happened. Once I did cut over, I was on a pretty different career path at the company, than I would have been if I'd stayed on the IC side.

I think the second was rather than fixating on one part of a ladder or level description, I think I generally took the approach of – I focused on always growing, I'm taking on harder or more complex problems and just delivering value. I think that just took me through the various stages of my career. Year over year, it was either shipping, or owning larger changes, or mentoring and developing people on different teams and in different roles.

[0:13:32.7] JM: In that time span that you were growing and accelerating as an engineer through the management ranks, this was around the time where other Facebook engineers I've

talked to have talked about a characteristic of this time where engineers who could build something, or lead something themselves and earn credibility in the organization were able to move upwards in a way that it sounded natural.

I think this is just a characteristic of an organization when it's at a certain smaller size, but it's in a hyper – the product is in a hyper growth state and there's this somewhat natural selection process by which people move up the ranks. Does that fit your recollection of the process?

[0:14:23.1] RY: I don't know if I would have thought it characterized it that way. You mentioned the phrase natural selection, I don't know that it was so limited, like only certain people who pass the threshold and made it.

I actually thought it was a great culture in that it really tried to recognize different forms of impact as people grew, and so what I was referring to earlier if you looked at your engineering director, or whatever level of engineer, there are many, many different types of people that would fit in that level.

I actually thought it was a pretty wide range of recognizing people at different levels. I think for me similarly, I don't know that there was any one thing that I did that helped me grow specifically. I actually switched teams a decent number of times at the company. I ended up working on many different parts of the product. Maybe that was part of it or for me, it was this more well-rounded experience working on some of the core full-stack products. Yeah, I wouldn't chalk it up to a specific thing that I did.

[SPONSOR MESSAGE]

[0:15:31.4] JM: When I'm building a new product, G2i is the company that I call on to help me find a developer who can build the first version of my product. G2i is a hiring platform run by engineers that matches you with React, React Native, GraphQL and mobile engineers who you can trust.

Whether you are a new company building your first product like me, or an established company that wants additional engineering help, G2i has the talent that you need to accomplish your goals. Go to softwareengineeringdaily.com/g2i to learn more about what G2i has to offer.

We've also done several shows with the people who run G2i, Gabe Greenberg and the rest of his team. These are engineers who know about the React ecosystem, about the mobile ecosystem, about GraphQL, React Native. They know their stuff and they run a great organization.

In my personal experience, G2i has linked me up with experienced engineers that can fit my budget and the G2i staff are friendly and easy to work with. They know how product development works. They can help you find the perfect engineer for your stack and you can go to softwareengineeringdaily.com/g2i to learn more about G2i. Thank you to G2i for being a great supporter of Software Engineering Daily, both as listeners and also as people who have contributed code that have helped me out in my projects.

If you want to get some additional help for your engineering projects, go to softwareengineeringdaily.com/g2i.

[INTERVIEW CONTINUED]

[0:17:23.2] JM: When you move into management, your job can become much more about human interaction and empathy than specific engineering problems. There's also an element of politics that comes into play more acutely. What adjustments did you have to make to optimize for that new set of roles?

[0:17:44.8] RY: My first switch into management was actually really quite gradual. The first team I managed was just three people, including myself. I remember the day-to-day really didn't look that different. I'd say over time, I wouldn't really describe it as shifting fully towards human interaction versus engineering, but it was a more depth versus – sorry, breadth versus depth approach. Where on the management side, I felt much more accountable for a wider range of problems, including people ones, but also technical ones.

A big part of making that adjustment was really accepting the new role. One example I always give is as an engineer and I think this is true of many engineers, I was a bit of a perfectionist. I thought a lot about things in black and white and designing systems. I'd be pretty fixated on really exhaustively listing out all the edge cases, figuring out that whatever I was building worked well. As a manager, it took time, but I realized you just can't solve problems that way. People are not systems, they're not optimization problems and you just have to accept there's a range of solutions and there isn't one true great answer.

I think a lot of it is just personal acceptance. One thing that I found funny when I look back on it now, I don't know if you put much stock into Myers-Briggs, but I was a lifelong Jay, so someone who saw things in black and white, was very prescriptive about how I thought about the world. After several years of managing, I actually shifted over to P. I think that's pretty descriptive and summarizes how I've changed how I think about people and systems.

[0:19:19.5] JM: What was your interaction with VP level employees, or the higher ranks of the hierarchy?

[0:19:29.5] RY: Yeah, one thing that was really great about the culture was how non-title driven it was. When you say the VP people, it actually takes me a second to think about who were the VPs that come to mind. I think, I did work with them for sure I think in different product review forums, like taking on new projects, goal-setting reviews and so forth. I also think VP level employees worked quite a bit just day-to-day with people across the organization. They could give feedback on pretty minor changes. If you run into them in the hall, they could chime in on different discussions. It was a very collaborative environment across levels.

[0:20:11.6] JM: What do you miss most about Facebook's engineering culture?

[0:20:14.6] RY: I do really miss that there was a very strong feeling of collaboration. I think, everyone felt you were part of a broader company, or mission and you're working on something together. That was a big part of it. On the technical side, I appreciate this much more years after leaving, but it's a uniquely innovative place when it comes to developer tooling and investing in infrastructure and efficiency.

I remember as an employee, we had these engineering all-hands every, I think quarter, or every few months. Every engineering all-hands, I would joke that the info team would get up on stage and they would just tell us again and again that everything just got faster and cheaper. I realize now that's pretty incredible if you think about the rate of growth, the fact that the underlying infrastructure just continued to get faster and cheaper.

I think that you really had this amazingly strong foundation of developer efficiency, developer tools that you could work on top of as an engineer. That's something that I think is pretty special.

[0:21:20.4] JM: There's a saying that I've heard from other people who have come on the show to talk about Facebook is that you throw out best practices that are yet less useful when applied in the scope of a high scale social networking product. Were there any best practices, or computer science, things in your tool belt that you threw out because of the uniqueness of the company, or that were accepted because of the uniqueness of the company?

[0:21:50.2] RY: There is one practice that I wouldn't say was good or bad I got thrown out, but I think changed quite a bit as the company involved. This was related to documentation. Coming in, you hear a lot about design docs and these extensive overviews of systems that you might write up when you're trying to launch a new product or rewrite some infrastructure.

In the first few years of Facebook, it was very light on that type of documentation. Instead, I'd say the bias was towards explaining context in the code, in individual changes, iteration, testing things live, at least internally iterating on it that way. Documentation was very integrated in development and there weren't as many standalone docs.

I think that changed over time. Early on, it was actually really great for the culture and I think efficiency. One example was we could be building some new feature on some part of the code base and trying to figure out how these decisions were made. You can actually just look at old previous code changes in place and rebuild comments from engineers from years ago and you would instantly understand what's going on. It’s basically this layer cake of documentation that you could wade through as you're making changes.

I think if we had used standalone documents more heavily, that would have been a lot harder and would maybe slowed us down in terms of iterating much more quickly. I think that's probably changed as the company has grown, but it's something I remember pretty well from the early days.

[0:23:25.8] JM: What did Facebook do to make sure that its culture was able to scale as the company grew?

[0:23:31.7] RY: Culture is it's such a funny thing to define. I feel that a lot of it comes through most strongly in everyday actions. Culture is preserved if you – as you work, you really understand how it relates to what you're doing. When I think of the engineering culture, I think this is very true Facebook, where it was very clear in every change that you made, so code that you wrote, you're testing, you're deploying, you're interacting with other engineers online or chatting about and testing features. I think that's where the culture really came through. Part of that was definitely the use of certain types of tooling and cultural norms around code review, iteration, comments that I think was – in some ways, easier to preserve because it was found in these smaller actions, versus some top-down imposed culture.

[0:24:24.9] JM: When you were at Facebook, you started and led the women in tech group. How is the workplace environment around diversity changed from your perspective when you started working in tech about a decade ago?

[0:24:41.5] RY: There are a lot more young people who are excited about tech these days, which I think is really great to see. There are just generations of new engineers, who both grew up using many of the products and services that 10 years ago, very innovative and new. They grew up on them, they really understand them to a deep level.

What I see is just a broader interest in tech and how it affects the world that comes from a lot of these young engineers. I think that's really exciting. I think that applies to if you look at the different backgrounds of new engineers and there are people today in tech who 10 years would not have become software engineers, and would have seen it as a much more limited role. You have people who came into it because of interest in other scientific fields, or even yeah, just completely different reasons.

[0:25:34.1] JM: You're now at Stripe. Stripe is at its core, a very different product than Facebook at its core. How do the cultures of the two organizations differ?

[0:25:46.1] RY: Something that's very true at both companies is this sense of being very mission-driven and having a very clear mission that you're working on. Something that I think is very exciting is if you ask someone walking down the street probably at either company, why do you work there? What is the company hoping to achieve? They'll be able to give you an answer. I think that's very similar and great. Of course, the actual missions and the products are very different.

Stripe mission is to increase the GDP of the internet. In many ways, it's a very economic, it's a very infrastructure-focused mission. As a result when you think of product development and building teams, there's an emphasis on building this API infrastructure layer that I think ties into how you think about your users, how do you design your products to in some ways, be more static so you can preserve all these different versions of the API at once.

With Facebook, it's a much more making the world more open and connected, it’s dynamic, it's about creating interaction and collaboration, so it ends up being a more iterative, collaborative environment.

[0:26:54.9] JM: Facebook was born in a time where you needed to roll your own infrastructure, you needed to build your own data centers and so on. Stripe is built in a time where you have cloud infrastructure. How does that affect the developments within the organization?

[0:27:14.0] RY: In many respects, it's not that different in that if you think about the engineering org, or even your system diagram internally, there are different components that we manage by different teams. I would say the big difference is a company like Facebook which has its own data centers and its own full-stack infrastructure, there will be entire teams that exist to go install new racks, or check on data center health, or do all of those things that you don't need at a company that's cloud-based.

I would say the big difference is there are just certain teams that don't exist. If you step back and you actually look at the interfaces between teams, the teams sit on top of the infrastructure, I think it's very similar. You still have teams that manage the cloud infrastructure, even if they're not building some of the low-level primitives themselves, they're still managing the internal interfaces that product teams would use to develop on top of.

[0:28:10.9] JM: At Stripe, there's a big emphasis on the writing process, like writing down your thoughts. This was something I also experienced at Amazon with the six-pager memo process that's quite popular there. At Facebook, was there any emphasis on writing, or was there an alternative emphasis perhaps on building product and showing your thoughts through code?

[0:28:36.8] RY: Yeah. I think iteration was a pretty big part of the Facebook development practice. There was focus on prototyping, iteration, interactivity. Being able to test something, being able to visualize something was really important. Yeah, so I think there was definitely a bias towards that.

[0:28:57.6] JM: There have been multiple books written about Google's engineering culture. I recognize you have not worked at Google, but I think in the formula days of Facebook, a lot of people were looking at Google as maybe a model for how Facebook could develop its engineering culture.

Facebook was distinguished from Google. Do you have a sense of how the engineering culture is contrasted, or what Facebook did deliberately different?

[0:29:31.7] RY: It's hard to say given not a ton of experience with Google. I did internet Google many years ago, but I wouldn't want to extrapolate too much from a single internship. A few differences that I'd say we're at least talked about, one was related to there was a bit more of a technical purism of Google.

One example was even when it came to code reviews, which are really a fundamental building block of development on a team. At Facebook, it had a really high bar for code reviews certainly. Every piece of code was carefully reviewed, there would be a lot of back-and-forth and just

making sure that the right tests are in place and everything worked properly. There was always that from the beginning.

What was interesting about Google is there are many layers of that process. I don't know if this is still true today, but 10 years ago, you had owners files where specific owners of every project had to be a reviewer. You had language readability where you had to be qualified reviewer for each of the languages, so C++, or Java, or whatever language you are working in. You added these layers that I think were in the name of holding very specific, technical bar, having this clear process about related to what does it mean to be technically proficient in each of these layers.

I think Facebook didn't really have any of that. It was much more around is the thing working the way it should be? Do we have the right checks and balances? It was much less focused on each of these sub-dimensions of breaking down the technical quality. That's one example. I'm sure it manifested in many ways.

I think the engineering ladders and levels were obviously very different as well, where Google has this – had a more, I would say, perhaps fluid range between an engineering IC and a manager and a tech lead. There would be people maybe performing – working across the spectrum, while I think at least when I was at Facebook, the two ladders were more distinct. While there's a lot of overlapping responsibilities, you were either people managing officially, or you weren't.

[SPONSOR MESSAGE]

[0:31:57.7] JM: There are so many good podcasts to listen to these days and it can be hard to make time to sit down and read a full-length book. There are more good books than ever. I like business books and self-help books and history books, but I don't have time to get through all the books that I want to.

Blinkist gives you the best takeaways, the need-to-now information and the important points of thousands of non-fiction books, condensed into 15 minutes that you can read or listen to. I like Blinkist, because I like audio and Blinkist is an innovative useful audio format. You can get a

free 7-day trial and support software engineering daily by going to blinkist.com/sedaily and signing up.

On Blinkist, I've listened to a few very long books about China in their 15-minute form. I also use Blinkist to review great books that I've read in the past, such as *Principles* by Ray Dalio, or *Being Mortal* by Atul Gawande.

Try out your free 7-day trial by going to blinkist.com/sedaily and signing up. That's blinkist.com/sedaly. Get those books condensed into 15 minutes and get more throughput in your book reading activities. Thanks to Blinkist for being a sponsor of Software Engineering Daily.

[INTERVIEW CONTINUED]

[0:33:36.8] JM: When you were doing management at Facebook, were there any procedures that you got into the habit of doing that you found particularly useful, like weekly one-on-ones, or doing something with your calendar, or something specific that stands out as a habit that really helped you out as a manager?

[0:33:56.2] RY: I learned a lot of the tactics while there, so things you described, like the weekly one-on-ones, or calendaring. One thing that I've definitely taken with me is it was a very cross-functional environment. There was a big emphasis on as an engineering manager on what you were doing, was it the right thing? Was it working well? Are the products that you were building the right ones and what was the user impact?

As a result, you had to work very closely with product managers and product designers and just people across the organization. We had a good practice of these, essentially the cross-functional meetings. You had a very, very tight communication and feedback loop with all these different partners.

I remember something I tell teams now is when you work with a product manager, it's a bit like a marriage; everything you're talking about, you should have some shared understanding of

everything that's going on with your product, your team, and how do you continue to communicate just almost on a daily, or an hourly basis at times to keep in sync?

That's something I've done for my team's is typically we have project-based, or team-based cross-functional group meetings, or group channels that we make quite a lot of use of, just so that everyone's on the same page and everyone's keeping an eye on what the end results of the work are.

[0:35:18.3] JM: Today, you're spending a lot of time in the APAC region in your work at Stripe. How does the Asian social networking market compare to that of the US?

[0:35:29.0] RY: You can definitely see the presence of the global players across the region. Facebook, Instagram, these are products that are used across the region as well. What's interesting is how they tend to use the products and the emphasis on messaging, or smaller group communities.

WhatsApp is very popular in different parts of the region, but so are other messaging apps, like Line, or WeChat from different companies across Asia. I think the big dimensions are there's pretty big melting pot of international and regional companies and are building products and people use a mix of them. I think there's a bigger emphasis on smaller group interactions and there are some products that are just very, very localized. They will have all of the instructions and settings only in a certain language, so maybe it's only in Chinese, which might make it a little difficult for people who can't read Chinese to use them. I think there are many products that look like that as well.

[0:36:32.2] JM: There was a long period of time where I think the conventional wisdom was that social networking was a winner-take-all market. Today, it seems much more like social is almost an API, or an application component that can just exist in a product. I think about the social networks I use on a regular basis. I mean, I've got Quora and Twitter and obviously, Facebook. I use Instagram some. It feels we're in the very early stages of social networking. Why did we think it was a winner-take-all market for so long? Do you have any beliefs on how the market will develop in the near future?

[0:37:15.6] RY: I don't know that I personally really thought it was a winner-take-all market. I think early on, I had actually used a few international social network products when I was in college. I thought a lot about the way that we message and the way that we have different communities for different types of interests. I don't know that I personally ever thought it was winner-take-all. It's interesting to see that I think you're right, the industry perspective has shifted.

One thing that's really interesting about living in APAC is you see that it's just not a winner-take-all world. There are so many regional differences. As I mentioned, there's language restrictions, there's cultural ones, there's just a desire for people to have different groups. I think this is actually not something new, it's just maybe something people are looking at differently.

[0:38:06.2] JM: One of the things that distinguishes the United States from at least China, and I believe this is true across Asia, is the willingness to pay for things using the internet. Obviously, some of this is due to the whole leapfrog effect and skipping the credit card and going straight to mobile payments. Is there also some cultural factor? For example, people pay for podcasts on a regular basis in Asia, which makes me very jealous.

[0:38:39.0] RY: It's real hard to comment, because I think it's so regional, it’s so country specific, or market specific. I mean, one thing I can see is how hard or easy is it to access that content, right? I think in the US and many markets, you get used to having an abundance of content that is free, and so that creates certain market dynamics, where you just expect things to be free, which is also a pretty vibrant ecosystem around ads and monetization in other ways.

I would say in some regional markets, there's just a shortage of content. You might be willing to pay for things that you see as premium, because it's harder to get access to them otherwise. One thing I found is just it's surprisingly still sometimes hard to find the right content.

One example is I talk to a lot of folks around here who sometimes ask, “Oh, where can I find great engineering management advice, or resources?” I think that's funny. I think if you live in San Francisco, you're used to seeing new thought leadership posts, or resources online posted basically all the time.

Here is just a little bit hard to discover that. There's people seeking for and I think would probably pay for it in a way that they wouldn't in San Francisco. I also wouldn't generalize too much. One example is in Japan, it's actually culturally still not that common to pay for things online. There's a degree of trust that's missing with online payments and people still in the majority, prefer to use cash or to pay after receiving a service in the deferred payment models. You wouldn't even speak about Asia, or the world. It really depends on the specific country, or market you're talking about.

[0:40:14.3] JM: Well, not to generalize further, but what have you learned about the Asian startup ecosystem that has surprised you? How does the Asian startup ecosystem contrast with the Silicon Valley one that you were so used to?

[0:40:30.5] RY: There's a really exciting feeling of growth and momentum in the region, where if you look at just the maturity of the markets and the companies, they're at a different stage. I think it's newer, it's really only in the last several years where you have bigger companies growing and building large engineering teams. I think it's a bit of a different maturity level, but I think the momentum and growth is really strong. It creates this really exciting energy, I would say in the region. Some things are very similar. I think people use cloud platforms, they use very similar development tools, you hear about React, you hear about GraphQL, you hear about GCP, AWS. People are really using a lot of the same tools.

I think what's different is there are fewer people with as much experience, or depth of experience in using some of these tools, because these products and companies are relatively new and coming online. I think there's a lot of enthusiasm and desire to learn from best practices in other parts of the world, but what's neat is they're looking at the whole world. I think, there's looking at the valley and enter and understand what works there. Is looking at China and India and the big mature companies that have done really well there, and this desire to learn from the best across the world.

[0:41:49.6] JM: One reason that you're in the APAC region is to improve the product of Stripe to the people who are using Stripe in the APAC region. One way that the Stripe product I think differentiates from the Facebook product is that much of what Stripe has to do is abstracting away legacy payments infrastructure from developers. You have to deal with some legacy

infrastructure. Whereas Facebook is to a large part, you're writing its own future and doesn't really have to integrate with anybody. How does that change your approach to engineering when you have to understand this archaeological dig of payment systems?

[0:42:36.0] RY: For one, there are a lot of engineers at Stripe that have to really understand how these partner systems work. I think there's a lot of time invested in really reading the specs, understanding how the API works and making sure the integration works really well. I think this actually happens at every company. There are many aspects of Facebook that also works with partners, or integrates other APIs. That's not that dissimilar. I do think it's part of the core service and product at Stripe in a way that's pretty different.

[0:43:09.6] JM: One thing that I have found is a trend among the most successful engineering organizations or startups, you might say technology companies, is the charisma of the founders. I've talked about this with investors who have come on the show also, the aspect of charisma and I guess the other words that you might associate with a good leader. What are the characteristics of the CEOs that you've interacted with? Mark Zuckerberg and perhaps, Patrick and John Collison that you have found particularly relevant in their ability to lead organizations for such a long period of time and continually be able to recruit and inspire people to work for them?

[0:44:00.9] RY: I mentioned earlier that the mission and being mission-driven is a big part of – a fabric of both companies. That really comes from the founders and from the CEOs in my opinion. I think that's something that I really notice and appreciate is how much people care about what we're doing and the mission and how it really ties into every decision the company makes, or everything that every employee is doing. That's a huge part of it. Bringing that passion to communicating the mission internally and externally is a really big part of it as well.

I think another one is detail orientation, just really understanding how the product works and thinking through the user, the edge cases, or how different changes would influence users. I think that's something that I've also seen to be very true in both companies.

[0:44:58.4] JM: Raylene Yung, thanks for coming back on Software Engineering Daily. It's been great talking to you.

[0:45:01.8] RY: Thank you. Thanks for having me.

[END OF INTERVIEW]

[0:45:07.6] JM: Commercial open source software businesses build their business model around an open source software project. Software businesses built around open source software operate differently than those built around proprietary software.

The Open Core Summit is a conference for commercial open source software. If you are building a business around open source software, check out the Open Core Summit, September 19th and 20th at The Palace of Fine Arts in San Francisco. Go to opencoresummit.com to register.

At Open Core Summit, we'll discuss the engineering, business strategy and investment landscape of commercial open source software businesses. Speakers will include people from HashiCorp, GitLab, Confluent, MongoDB and Docker. I will be emceeing the event and I'm hoping to do some onstage podcast-styled dialogues.

I am excited about the Open Core Summit, because open source software is the future. Most businesses don't gain that much by having their software be proprietary. As it becomes easier to build secure software, there will be even fewer reasons not to open source your code.

I love commercial open source businesses, because there are so many interesting technical problems. You've got governance issues. You got a strange business model. I'm looking forward to exploring these curiosities at the Open Core Summit and I hope to see you there. If you want to attend, check out opencoresummit.com. The conference is September 19th and 20th in San Francisco.

Open source is changing the world of software and it's changing the world that we live in. Check out the Open Core Summit by going to opencoresummit.com.

[END]