EPISODE 647

[INTRODUCTION]

[0:00:00.3] JM: Stripe is a payments API that allows merchants to transact online. Since the creation of the payments API, Stripe has expanded into adjacent services, such as fraud detection and business management and billing. These other verticals leverage the existing customer base and infrastructure that Stripe has developed from the success of that core payments business.

Raylene Yung is the Head of Payments at Stripe and she joins the show to talk about her work, which includes elements of engineering, product development, design and management. All of these dimensions of her job came up in our conversation and it made for a wide-ranging discussion. This interview comes in the context of Stripe's rapid growth.

The organization is changing and Raylene explored the questions that Stripe is asking itself internally about org structure; namely, what's the trade-off between a defined hierarchical structure of direct reports, versus a decentralized flat org structure? Raylene came to Stripe from Facebook, which is a famously decentralized flat org structure.

Another question we explored is the subject of highly defined roles versus less well-defined roles. In the software industry, you commonly come across people who are titled senior infrastructure software engineer, or SDE2. These are pretty well-defined naming conventions that exist in a ladder of different hierarchically defined roles, and it's up for debate whether those are titles that make sense, or if it's better to have people just have fluid roles, like designer, or software engineer, because then their stratification in the organization is less well-defined and their position in the organization is less well-defined, which can be good because maybe you want people to have fluid positions in the organization and have them self-assemble.

These structural company questions are thought-provoking. The fact that Raylene was willing to address them as open questions, I found that to be humble and I found it to be a useful discussion. If Stripe is having these questions and certainly there are lots of other organizations that are also having these kinds of questions. It's a really enjoyable discussion.

Before we get started, I want to mention that we're hiring a creative operations lead. If you are an excellent communicator, please check out our job posting for creative operations at softwareengineeringdaily.com/jobs. This is a great job for someone who just graduated a coding boot camp, or someone with a background in the arts who's making their way into technology.

If you want to be creative and you want to learn more about engineering and you like Software Engineering Daily, check it out at softwareengineeringdaily.com/jobs.

[SPONSOR MESSAGE]

[0:03:15.3] JM: Citus Data can scale your PostgresSQL database horizontally. For many of you, your PostgresSQL database is the heart of your application. You chose PostgresSQL because you trust it. After all, PostgresSQL is battle-tested, trustworthy database software.

Are you spending more and more time dealing with scalability issues? Citus distributes your data and your queries across multiple nodes. Are your queries getting slow? Citus can parallelize your SQL queries across multiple nodes, dramatically speeding them up and giving you much lower latency. Are you worried about hitting the limits of single node PostgresSQL and not being able to grow your app, or having to spend your time on database infrastructure instead of creating new features for your application? Available as open source, as a database, as a service and as enterprise software, Citus makes it simple to shard PostgresSQL.

Go to citusdata.com/sedaily to learn more about how Citus transforms PostgresSQL into a distributed database. That's C-I-T-U-S-D-A-T-A.com/sedaily, citusdata.com/sedaily. Get back the time that you're spending on database operations. Companies like Algolia, Prosperworks and Cisco are all using Citus, so they no longer have to worry about scaling their database. Try it yourself at citusdata.com/sedaily. That's citusdata.com/sedaily.

Thank you to Citus Data for being a sponsor of Software Engineering Daily.

[INTERVIEW]

[0:05:00.3] JM: Raylene Yung, you are the Head of Payments at Stripe. Welcome to Software Engineering Daily.
[0:05:03.5] RY: Thank you.

[0:05:04.4] JM: We've been doing a lot of shows lately on some different FinTech companies. We've talked to Klarna and N26 and just some other payments related, FinTech related subjects. I'd like to start at a high level. A developer who integrates with Stripe, that allows them to accept payments through their product. Walk me through the life cycle of a payment that gets paid through Stripe.

[0:05:33.0] RY: It's almost hard to answer that question simply, just because Stripe actually can do so much for our users. We do a lot more than just allowing you to accept payments and maybe, it won't be too expansive but the idea with Stripe is it enables you to essentially participate in the global economy very quickly, and we we're supported in 25 countries. It's not that straightforward, like every – depending on the country, the payment method that you use, or your customers based and where you're based, the answer is actually quite different. I'm happy to try to reduce it down, but just as a disclaimer, it's actually pretty complicated just because we can do so much more than your typical payment processing company.

[0:06:10.5] JM: Yeah, sure. Please do reduce it down to whatever level of abstraction works for you.

[0:06:15.3] RY: Great. The way we think about Stripe is we provide a set of composable products up and down the stack. To just start, there's an option from one of our users to figure out how do you want to accept payments? Do you want to build against their API directly, so that you maybe are setting up your own back-end, you're sending a request to Stripe API that bundles up charge data and sends it directly.

On the other hand, you could also choose to integrate against our different front-end components. You can choose checkout, which is a full web page that enables – we just serve an entire webpage for you, your users just type it in, we handle everything from there. Or you can do something in between that's using a set of front-end components. Essentially the idea is a

developer would choose the way they want to integrate with Stripe, expose something to the user to collect payment information.

Stripe actually handles the tokenization and secure – we basically are able to store all the sensitive data securely, so that our end users never have to actually touch the sensitive card data, or enter PCI scope. We handle all of that. We receive let's say a request to process a charge. Depending on the payment method, there are different ways that this can happen. A typical credit card transaction will just happen synchronously, so we'll send a request on the back end, basically authorized to charge and give you a response one way or another. Was it decline? Did it go through? Was the value – For some payment methods, they can either take you through a redirect flow, or maybe it's a more asynchronous transaction, like a debit, like a bank debit method.

From there, we have this transaction on Stripe. What's great about our product is we handle all the complexity in the backend. We'll take care of getting the money into your account and potentially taking care of currency conversion, or moving it into different accounts that we have on our end. What the user sees is just this transaction log; you can look at what are all the transactions, what happened to it, when does it land in my balance, and when do they get paid out? There are lots of different ways that we can help you configure that as well.

One example is our marketplace product connect; so that enables you to – actually, have a few more options on how you handle that transaction. One example is Lyft, where it's not so straightforward as a passenger on Lyft. You take a ride and you obviously are paying Lyft, but you're paying for a service by specific driver. What Stripe helps Lyft do is both model that transaction as a Lyft charge for driver Bob, and representing that transaction as both a transaction for Bob, but also for Lyft, and we help you reconcile that on the backend and also get the driver paid out very quickly.

[0:08:56.8] JM: From an engineering perspective, one thing that seems luxurious about working at Stripe is that the products that you're developing, the typical user is a developer and at many companies, you're developing a product that is not targeted at developers. I know Stripe is not explicitly an only producing products for developers, but the vast majority of the users are developers. You can assume a high-level of technical willingness, but also there is a focus on

simplicity, so that you have the sweet spot of product development. Also, the fact that you're building products for engineers and you are an engineer yourself, it becomes a little bit easier to empathize with the average user as opposed to if you're building a product for the general populace. I'm wondering how that affects product development and engineering from your point of view.

[0:09:55.1] RY: I think one thing is Stripe definitely started as a very developer-centric product, and I think it shows in a lot of our documentation and the product itself, the API design and even how our dashboard is designed. Over the years, I would say that while we are great product for developers, we also have users of many different kinds, and you alluded to this. I think, we also think of the example I just gave, we also think of the Lyft driver as a Stripe user in many ways, because they're providing a service and we're able to help them get paid.

I think we have actually a pretty diverse range of users. When it comes to product development, I think what we've done is we tend to start with a technical foundation. Like you said, it's very developer-centric, it's very composable, but we do from day one keep in mind how a variety of users might use it.

One example is earlier this year, we launched a big refresh to our recurring billing product, called Stripe billing. We started, like first principles, like developer-centric, we designed a set of easy building blocks in the API that interop really well. We made sure they integrated well with existing objects, so we added a few new ones, but we also looked at some of our oldest API objects, like customers and made sure they worked seamlessly with recurring billing. That was the foundation.

We also know that there are a variety of types of recurring business owners and users who may not want to immediately go to code and might want to test out one subscription plan, or one new invoice, or one product. From day one, we invested a lot also in building dashboard tools and we found that the dashboard tools maybe are dual purposes. They help users that aren't maybe just developers use the product very early, but they can also help developers test things out quickly and get started. We've gone on from there, where even on top of the dashboard tools we're building higher-level products, so you maybe can automate invoice delivery and

conversion without writing any code. I think we are moving up the stack over time, but we never lose that technical foundation.

[0:12:03.4] JM: When you build a payments API, you're abstracting away from the developer – things that a developer doesn't want to have to think about. I've been trying to understand the payment rails of the online economy a little bit recently in some recent shows. It's a little hard for me to understand. It at least feels in some ways foreign. You've got these things like credit card processors and/or I guess, I should say credit card companies, you've got banks. The minutia of the lifecycle of a payment as it goes through those parties is not necessarily intuitive to a developer, I think. At least it's not intuitive to me.

I'm imagining that my payment, when it goes through Stripe and under the covers, it's making its way through COBOL code and some weird old legacy infrastructure on the side of the credit card processors, or on the banks. How much of that stuff, as an engineering manager at Stripe, how much of that stuff do you have to understand? Do you have to know the guts of the legacy? Do you have to unearth the archaeological dig of the of the payments of the past?

[0:13:23.4] RY: Definitely. I think what's interesting about Stripe is how we can abstract that away from users. Let's say you're integrating with Stripe, you don't need to do that, and we actually – Our efforts are twofold; on one hand, the product side. We actually invest a lot in designing APIs that really try to abstract that away for external users and internal ones. One example is our payment method API V1 sources. What we try to do there is we take a range of payment methods that have different minutia, like you described. Settlement timing, or refund dynamics, things are very spec specific. We actually try to reduce them down to an internal API that simplifies it, so there's a source of money, there's a receiver of money, you can adjust the timing and so forth.

We try to build that internally as well, so some teams can actually work on top of that in the same way and they benefit from that clarity and simplicity, the same way external users would. Of course, you go lower in the stack and there are teams at Stripe and payments that really do have to understand the nuts and bolts of the specs, and exactly what's happening with our financial partners and how we participate in the network. You will have some teams at Stripe

who reviews the specs and really try to understand what the latest is in – what our financial partners are doing and how to think about the different interactions of payment methods.

I think we try to take an approach, so that not every engineer in payments needs to understand exactly how the business spec works, and we try to build various different layers of internal infrastructure to make it easier and easier to understand higher up the stack you go.

[0:14:56.6] JM: When I accept payment online, maybe it comes from a credit card or an automated Clearinghouse payment, or Alipay, depending on the payment, what's the acceptance process look like on Stripe, and from an infrastructure perspective? Is it hitting some monolith? Is it hitting some payment identification gateway and then it tees off to the Alipay processing system, or the credit card processing system? What's that service to service communication sequence look like?

[0:15:34.5] RY: It mirrors what I just described. There's essentially an API layer, like an internal API layer. We might receive a bundle of data from for a specific transaction and we'll see, is it a card? Is it a Alipay transaction? Is it a bank debit method? There's a layer internally that we think about with cards and sources, so the same thing I describe, we call it V1 sources externally, but we also have sources served internally that knows how to process different payment types.

I would say there's a mirroring of the outside. It makes it into our API, we'll have different code paths depending, or even different services depending on the type of payment method and it goes down from there. Yes, it varies very widely. There are some payment methods that may go to even different partners on the backend depending on the country, payment method and payment method type.

[0:16:24.2] JM: Do you have to identify the latency of the different payment providers in order to figure out how to think about the external dependencies and what you're going to make asynchronous, etc.? What are the challenges of integrating with the various external dependencies?

[0:16:42.5] RY: I think the same that would apply to just any maybe product that has to manage a lot of external integrations. I think, there's some amount of having internal sense and reconciliation of what do all these external APIs look like. Again, we take a very scaled infrastructure approach. Let's say, a set of our partners, typically there's a lot of async batch filing, so you might aggregate transactions into a batch file, which we send and over the wire to specific partners. Each of them may have a different spec, so they might require different fields and different orders and so forth.

What we've done is we have to understand that of course in order to give them the correct data, but we also try to abstract that internally, so we can build an internal batch infrastructure of API translation layer, so that we can scale and more easily plug in new partners.

[SPONSOR MESSAGE]

[0:17:39.8] JM: Cloud computing can get expensive. If you're spending too much money on your cloud infrastructure, check out DoIT International. DoIT International helps startups optimize the cost of their workloads across Google Cloud and AWS, so that the startups can spend more time building their new software and less time reducing their cost.

DoIT International helps clients optimize their costs, and if your cloud bill is over $10,000 per month, you can get a free cost optimization assessment by going to doit-intl.com/sedaily. That's D-O-I-T-I-N-T-L.com/sedaily. This assessment will show you how you can save money on your cloud, and DoIT International is offering it to our listeners for free. They normally charge $5,000 for this assessment, but DoIT International is offering it free to listeners of the show with more than $10,000 in monthly spend.

If you don't know whether or not you're spending $10,000 if your company is that big, there's a good chance you're spending $10,000, so maybe go ask somebody else in the finance department.

DoIT International is a company that's made up of experts in cloud engineering and optimization. They can help you run your infrastructure more efficiently by helping you use commitments, spot instances, right sizing and unique purchasing techniques. This to me sounds

extremely domain-specific, so it makes sense to me from that perspective to hire a team of people who can help you figure out how to implement these techniques.

DoIT International can help you write more efficient code, they can help you build more efficient infrastructure. They also have their own custom software that they've written, which is a complete cost optimization platform for Google Cloud, and that's available at reoptimize.io is a free service, if you want to check out what DoIT International is capable of building.

DoIT International are experts in cloud cost optimization. If you're spending more than $10,000, you can get a free assessment by going to doit-intl.com/sedaily and see how much money you can save on your cloud deployment.

[INTERVIEW CONTINUED]

[0:20:03.5] JM: The fraud detection and risk mitigation process is core to every payments company that I've talked to, every e-bank, or challenger bank that I've talked to. They spend a lot of time thinking about fraud and risk. I did a whole show with Michael Manapat, I think it was a year and a half ago or so, where we talked about, that we talked about radar and the data engineering, the data science behind that. I think organizationally, that product set falls within payments. How much time do you spend thinking about fraud and risk, and what's the – in terms of the hierarchical management structure, how does that interaction between you and Michael for example, or you and the risk team, where's the partitioning of responsibilities there?

[0:20:55.7] RY: Yeah. We're very closely with Michael. I think one thing just to say is we've been growing extremely rapidly over the past few years. I think it's something that's been incredibly exciting to see. Stripe when I joined was just over 200 employees, and today we're over a 1,000. That means we've also had to scale our engineering in a variety of ways. I guess, what I'll say is I actually don't think the way the org is just necessarily structured is that important in any given point in time. It's really how teams work together and how do we build shared metrics, shared goals and shared interfaces.

I mean, to answer your question, I do work very closely with Michael and actually quite directly, given how important and central fraud is to payments. I'd say the way we think about what you

describe how do you think about risk, so I think we think about payments from a very product-centric view, or as I describe we have this layer of financial infrastructure, we have internal abstractions, we have different product verticals, and that's how we set things up.

Stripe is an incredibly, in a very exciting way, very cross-functional, very collaborative. When you think about how does risk work with payments, we have shared missions, we've shared goals, so we might together look at fraud and risk metrics across the company and the shared meetings and set shared goals.

[0:22:11.8] JM: I don't know if you ever seen this diagram of the different company structures, where they have the graphical structure of the company and the hierarchical structure. You take Amazon, it's purely, it's like a binary tree and it's two people answer to the top person and two people answer to each of those people, and you know exactly who is your direct report and who you are reporting to. They also graph Facebook and it's like a directed – it's a completely directed acyclic graph. It's supposed to be modeled as completely decentralized.

Those set of graphs is always stuck in my head as something I study, because companies are successful with different orgs structures and you also see these different org structures break and fracture with different volumes of employees. I mean, can you give me more of a perspective for how Stripe organizationally as you're growing, I'm sure you're having these kinds of conversations, how hierarchical you want to be, versus how decentralized and flat-ish you want to be?

[0:23:18.3] RY: I've definitely seen that graphic. It's pretty funny. I think that's actually very reductive. I think, we do think a lot about how we scale across engineering teams and how the structure should be. I think, in general – I think, I spend a long time in my career working on scalable infrastructure and scalable platform products. I think, scaling and engineering org is very similar. You want to be pretty responsive to changing needs of the organization, but you also want to be pretty honest and take a look at what's working what's not and iterate on it as you grow.

I would say Stripe tends to be a pretty, I guess, I wouldn't say an empowered, a fairly, in some ways decentralized from a hierarchical perspective. It's not like we have a very deep tree, or

anything like that, but we actually – idea is for each engineering team, what is the mission? What is the goal? How they work together with the teams around them, is the highest order. We also think a lot about healthy growth and reliable and scalable growth. We think a lot about when we look at a certain structure, what are the challenges with it when it comes to how it impacts the business. I'm sorry. It's a little bit hard to – basically, it's hard to answer your questions. I don't think we really fit one of those diagrams.

[0:24:34.6] JM: I want to say by the way, I think that's fine, because also taking overly over-fitting on the past, past companies, or companies that are still thriving today with their internal org structure they developed five or six years ago, you can't closely fit your organization to it, partially because there have been innovation since then. We didn't have Slack widely accepted 10 years ago. We didn't have Asana widely accepted, or I don't know what your project management system is, but Asana is a little bit different. You could argue Twitter is a collaboration tool that everybody's on. Things have changed, so you have to reorient yourself in time. There's a lack of precedent to orient yourself on.

[0:25:17.1] RY: Actually that's a great point. Maybe what I would say is what characterizes our engineering or growth, I would say is being flexible to change. As we've grown, so I mentioned I've seen the company already grow from 200 to a 1,000, and just super excited for the growth we have ahead of us. Something we've done is continuously iterate and adjust for scale. I can think of a few examples.

One example is when I joined the company, as you mentioned, we had a very API developer-centric product and culture, so we had a lot of API generalist engineers, and had relatively few front-end specialists, relatively few machine learning specialists, but as we've grown – and as a result, we had a very centralized front-end team. We had a couple front-end engineers who would support the front-end products and they were to serve on their own team.

As we grew, we saw that we needed to scale. We're building many, many more front-end web products, and so we started building a front-end hiring pipeline, we started forming a larger and larger team. When we saw that scale, we realized, actually it made more sense to have front-end engineers embedded closer to their product areas to, as I mentioned, better group around the mission and goals and the business impact that we were trying to have. Today, we actually

have dozens of front-end engineers that are throughout the engineering org embedded in different teams, but just two years ago we had only one centralized team.

Another example that's very top of mind for me that demonstrates this flexibility and scaling mindset is our approach to global expansion. I think for a long time we would have people on larger teams focused on global expansion. Within let's say, a part of our financial infrastructure stack, we might have most of the people working on just shared infrastructure, or scaling up our largest partners and we have a handful integrating new international partners. That worked for a long, long time. As we've seen today, with our very ambitious global expansion goals, we've actually broken ground on not only having a more dedicated engineering team for global expansion, but we've actually for the first time set up an engineering office outside of the US. Just earlier this month, we sent our initial landing team to Dublin to kickstart global engineering in a much more decentralized distributed way of working.

[0:27:35.4] JM: Do you also rethink the role, so like the traditional set of roles at an organization that makes software technology products is like program manager, project manager, may be QA person manager, designer, software engineer, SDE2, SDE3. It's like a menu. You go to a McDonald's and it's always the same menu, and there's certain companies where it's like, they're just going to copy that model. That model works in many ways, but I feel there is maybe a bit more subtlety to the roles at Stripe.

Stripe is obviously designed first, for example, which it's curious. Like the whole design and the focus on design and developers, even though an engineer typically when they're working with an API it's like, "Okay, what matters to me is the design of the API? Is it a nicely formed JSON object?" Actually, developers care a lot about the visuals too, which is an underestimated thing that Stripe has taken advantage of. Yeah, role fluidity. What do you think about roles and new roles? Should people have these very traditional distinct roles, like program manager, or should it be – or is there something different, more subtle?

[0:28:54.0] RY: Well, it's something you were saying earlier. I think the broader just ecosystem, or tools that we can use in software engineering are continuously evolving and changing. One thing comes to mind is even 5-10 years ago, you would have large teams of people who had

managed data center operations, right? Today, that's just not a skill set that Stripe needs. We build on the cloud.

I think as you're saying as these different productivity tools, or even just different software paradigms emerge, roles will naturally need to evolve and change. I think Stripe is unique in this regard, because I think we really push for a flexibility and an adaptive nature of our engineering team. What we have is we have a really strong culture of internal mobility. We've actually had engineers move from infrastructure and security teams into core product teams and vice versa. We've had people move from more like front-end, backend, mobile to AI and so forth. I think that shows that the growth mindset that we have, it also I think creates a really interesting and strong engineering culture, because it feels like people can learn and they're not pegged into very, very specific roles.

For example, we actually just have a software engineering track today. We don't really have like, I don't know, some companies like you said, go off the menu and they might have DevOps engineers, site reliability engineer, these different sub-flavors. For us, we really believe in a unified very strong engineering culture.

[0:30:21.0] JM: What about the constraints? What constraints do you want to put on your teams, the engineering teams? Obviously, the fluidity is great, but you probably don't want every programming – you don't want people writing code in some COBOL in Stripe. You don't want to give that level of freedom. You probably want to standardize on a continuous delivery tool, or maybe not, because there's so many of them today, but you want to standardize – if you're on AWS at least, I don't know if you're on – if you're also going multi-cloud with Google, but do you have platform engineering? Do you have standards, the constraints that you put on developers?

[0:30:56.7] RY: I think we definitely believe in investing a lot of developer tooling and productivity. I think that's where the standards you might come from. I don't know that we think about them really like limitations. I think it's really a flip side, like what can we do to help people move faster? In the same example, in some cases adding a new language to build one new service is not the fastest path to getting impact, right? I think we have a very users-first impact-driven environment, and that's where a lot of our decisions come from.

I think the reality is yeah, we do believe it's faster in many ways to use existing languages, or existing infrastructure in the way that we set up certain services that enables if a new team coming along enables them to get up running – get up and running much more quickly, so that tends to if there are some standards that emerge that way. They're mostly driven from the end result and not from a limitation, or a restriction.

[0:31:49.9] JM: That's great. You don't have to do the these are the four blessed languages that you may use at Stripe. You don't have to do that thing?

[0:31:57.2] RY: I mean, we've talked about that a lot, but in the way, I think, let me put it this way, even if we – let's say today we have a couple, a few blessed languages, I think if there was a new business need that we learned about that made us rethink that and said, “Hey, we actually you know, let's say we're using Ruby a lot for this, but this problem is just so different, we need to use a different language, it would totally do it.” In fact, this actually happened earlier this year, we acquired a company that does point-of-sales products that we’re really, really excited about.

You can imagine like, point-of-sale, there's a hardware component, there's so many different things that are very different from processing just online payments. That has pulled in a few different languages, a few different technologies, because we see that it's going to make a meaningful difference in the quality of the product.

[0:32:44.2] JM: What are they using for the hardware?

[0:32:46.3] RY: C++ is a lot of the – on the embedded side and Java and C++ are the main languages. I think we are still working on integrations there.

[0:32:54.8] JM: Interesting. That company that was not a super old company, right? I'm actually surprised they used C++ instead of roster, or Go, or maybe there's something about embedded systems that you still want C++ for.

[0:33:10.3] RY: I think a lot depends on the partners and how you're working with partners. In some cases, you're just integrating against building SDKs, or integrating against partner

software. There is some amount of just working with the ecosystem that influences some decisions there.

[0:33:22.7] JM: Are you standardized on any – I mean, I know you're standardized on AWS, but has there been any desire to like, "Yeah, I want to be on Google because of GKE, or because of the BigQuery, or something." Any desire to go multi-cloud for particular APIs?

[0:33:40.1] RY: I think the same principles applies. We're always looking and we always are keeping up with the latest and trying to see what's out there, because if there's an opportunity to meaningfully improve the quality of our services, our product, we might do that. I would say we're always looking and learning, but that's probably the main thing.

[0:33:55.9] JM: I actually have a question, it's not really super related to engineering, but more about business level productivity. We talked about Asana and Slack earlier, there's these other business software tools, things like front and Airtable, I see these companies raise a lot of money, and so I assume the tool is pretty useful and I'm actually tempted to try out ditching Google sheets and trying out Airtable and seeing what it does.

I guess, I haven't been tempted enough to actually try it out, but what do you think of this next generation of business productivity tools? Does that affect your engineering at all, or do you see those being used at Stripe? Do people experiment with those?

[0:34:42.0] RY: I have a lot of personal interest in tools, software tools. I play with a lot of them for my personal projects. I think for Stripe, again our main focus is really building the best products as quickly as we can and maintaining a high-level of quality. I think it goes back to that. If a tool for some reason we saw would have a meaningful and may cause any meaningful improvement to our productivity, we would explore it. We've done that. We've upgraded various things we use internally, but I don't think we have an objective of just playing with tools or really trying to find the next best one.

[SPONSOR MESSAGE]

[0:35:22.1] JM: Failure is unpredictable. You don't know when your system will break, but you know it will happen. Gremlin prepares for these outages. Gremlin provides resilience as a service using chaos engineering techniques pioneered at Netflix and Amazon.

Prepare your team for disaster by proactively testing failure scenarios. Max out CPU, black hole or slow down network traffic to a dependency, terminate processes and hosts. Each of these shows how your system reacts, allowing you to harden things before a production incident.

Checkout Gremlin and get a free demo by going to gremlin.com/sedaily. That's gremlin.com/sedaily to get your free demo of how Gremlin can help you prepare with resilience as a service.

[INTERVIEW CONTINUED]

[0:36:21.0] JM: Have there been any particular tools that have been exciting to you that have come out recently on the engineering front? Maybe there's a CD product, or a new API that came out that's solved some problem? Any, or some – a new AWS service, or anything recent that's really taken you by surprise?

[0:36:42.1] RY: I don't know that we have – I don't think this is something that actually exists, but something I think a lot about is how we're going to scale and expand globally. I see, there's a lot of conversations happening now with major cloud providers about how to – how can they successfully help people expand globally with a higher level of security, data encryption, data locality? I don't think there's actually anything out there yet, but I'm definitely on the edge of my seat wanting to see what happens there.

[0:37:09.8] JM: Yeah. This is an interesting problem, because I think I talked to somebody who was talking about going multi-cloud, or going across the world and they're trying to replicate a Kafka cluster across the world. That's hard, right? Replicating a really big distributed queue across the world, that's – I don't think that's a solved problem.

[0:37:31.0] RY: Yeah, definitely.

[0:37:32.5] JM: Are there any particular engineering pains around going international that you've found problem? As like latency, or replication? What is it exactly that makes international hard?

[0:37:42.7] RY: I definitely think latency, things where you've made an assumption about being in one location and it permeates your product, that makes it really challenging. I would say another is just local – the world is quite fractured when it comes to payments worldwide. Regulatory concerns, compliance restrictions, they're just very, very different. Something we've seen is in the same way that engineers might have to deeply understand Visa specs. When we enter a new country, we have to really understand what are the requirements there, how do the local payment methods work, what are the different terms and how what is the payment experience like for users?

That actually ends up being probably one of the hardest things. One example is with 3-D Secure, which is a new – essentially a payment standard, the trace increases security, reduce fraud in Europe. I would say from a US-centric view, we think of credit card payments, you type some things in, you click send and it – you actually want a faster process, you want it to work quickly, you just want to get through your shopping cart. For European users, they're actually accustomed to a heavier weight flow. It might redirect them to another site, so that so that they can better authenticate it. For that audience, that redirect and that heavier weight flow is actually better, because it increases their confidence in the merchant and in the product.

That's something that's really hard for us, because we – on one hand, we're trying to make our front-end flows maybe really fast, really efficient, really high-quality for certain users, but then we're like, actually they want a more intense redirect based multi-step process. Those kinds of things can be very challenging.

[0:39:16.8] JM: In India, you have the cash-on-demand thing, right? Is that something Stripe has tackled?

[0:39:22.2] RY: We have not been – We're not working on that right now. I would say, it's something that we're learning a lot about. It's not just India. I think many countries in emerging

markets in Southeast Asia and Africa, cash is a huge component of their – of yeah, of payments. I think that's just something we're learning a lot more about and thinking about how to support.

[0:39:41.4] JM: Yeah. What are the problems of – how would you make a – have you batted around this idea, like how can we make an API for cash-on-demand?

[0:39:49.5] RY: Actually, so what's interesting is I think – as I saying, I think we think a lot about how do we build common infrastructure tools and language around payment methods? As I mentioned, we are building a lot more in card-present, supporting card-present. There's something very analogous there. You have a different payment method, there's something that happens in the physical world that you have to do, and then the killer feature is how does that integrate seamlessly into your online business?

We do think about that a lot. I think, these APIs are the answer again. You maybe have some API layer, which we provide through V1 sources and our charge API that enables you to give us data somehow. What's great about Stripe is in the backend, we can use that data, create a canonical view of your business. I think cash is no different, or paper checks, or many of these different methods that seem older school. In the end, it's really a bundle of data that you're trying to give to – you're trying to move the data from a customer to a merchant, and in the end, the merchant just wants to get the money at the other side of that.

I think they're actually very common and I think something that's really exciting that we do and really try to reduce these complex problems down into something that is more simple and more composable.

[0:40:58.3] JM: I did a bunch of shows about cryptocurrency stuff a while ago. One thing that cryptocurrency stuff boils down to is uncensororable payments. That's what people – that's one thing that people really value out of the blockchain, like whether that is actually possible right now is up for debate since the ledger is centralized, I think in some particular mining organizations that could actually do a 51% attack if they wanted to. It's an open question as to whether you can actually have uncensorable payments today.

Conceptually, that's what cryptocurrency companies want to be. Now, censorship gets a bad rap, but in some cases you might actually want censorship. You might want certain types of organizations, or certain types of businesses to be "censored." If for no other reason, than to adhere to government standards that oversee a particular type of business; a business in the United States, for example you can't – you can't have an API for well, name your miscreant behavior. At least, that's not on the blockchain, or that's under scrutiny like Stripe. How do you think about the correct level of what to allow through Stripe's API?

[0:42:14.0] RY: I think short version is the teams I support, we really think a lot about building APIs for collecting the right – collecting information and processing payment and supporting transactions through Stripe. I would say there is another team kind of to echo [inaudible 0:42:26.6] point, that thinks a lot more about policy, so I would say that's a risk compliance. I do think that's actually a really interesting topic, but I'm probably not the best person to speak to that.

[0:42:37.2] JM: I completely agree with you/ That's totally fine. Raylene, you've been at Stripe for three years. How has your job changed in that period of time?

[0:42:44.8] RY: Really quite a bit in many ways. I'd say, as I mentioned it's just grown a lot, so I think when I started the product team that I led – it was really interesting. We had product team that built all of our products and was maybe 15 people, and there's one product manager started the same day. It was a very lean team. I would say over the few years, we've had to scale that up and we've launched many, many new products and we've grown the engineering team considerably. I'd say the way that I spend my time there today has changed a lot, but also just the demands of the engineering org and our different areas of focus as a company have also evolved.

[0:43:19.1] JM: As a manager in a perfect world, if you were doing things by the book, you would want to delegate everything. You want to spend as much time as possible thinking about higher level concerns, but there are managers that I've talked to that have emphasized the importance to me of occasionally signaling that you're capable of doing lower-level work. For example, you're capable of writing code. How do you look at that signaling? Do you look at that as important at all? Is it signaling that you're a manager, you're an engineer too, you're not just a

manager, you're willing to dig into the code, or do you feel like that's just – that's totally beyond your purview and that should be abstracted away from you?

[0:43:59.2] RY: I think you used an interesting word; the word signaling. Let me share just my take. In my first four to six weeks at Stripe, I actually did write some code. I pushed a small change to the way we displayed fees and in a certain region of the world. I wrote some code. I got familiar with the dev tool chain, and really wanted to deploy some actual production changes to our users. For me, it wasn't important from a signaling perspective. Truly it's about, it was about building empathy and some very personal understanding of what the day-to-day looked like for an engineer at Stripe.

I think there's a tremendous amount of value in that, but you can get it in different ways. Another recent example is I wanted to get a feel for what does it look to be on our weekly on-call rotation. I don't think it necessarily is better for anyone for me to actually be trying to debug issues in real-time for our users, but I did want to see what it was like. I shadowed an engineer and I sat next to him and watched as he went through very specific workflow details, like checking on specific bugs, examining the code and so forth.

I think for that – that can prevent – sorry, provide tremendous value as an engineering leader, because ultimately in the end, I think you're trying to support the health and happiness and productivity of your team, but you're also trying to represent to others what are the critical problems, what should we be doing to help the team? I think it's hard to do that without some degree of detailed knowledge. I don't think as a signaling, or symbolic value it's that important.

[0:45:36.3] JM: Okay, cool. Well just to wrap-up, what is something that you know about engineering management today that you wish you knew a year ago?

[0:45:44.1] RY: I think I've learned a lot about adaptability. I actually think that's something I've thought about throughout my career, but in particular in the last year, there's just how applicable adaptability is to a really wide range of problems and teams, I think is what I've learned. I think it's kind of, as we were saying before, things are always evolving and the world around you and day-to-day within your teams. There's something around as an engineering manager learning to adjust to change, almost actively be a participant in thinking about how to evolve your team to

the next chapter. I think that's incredibly, incredibly important, particularly in high growth environments, where you just have to understand that things are going to be different in three months than they are today.

I think the best engineering managers and leaders are able to not only be okay with that change, but to actively anticipate it and design everything around this idea of adaptability and scale.

[0:46:42.5] JM: Okay. Well Raylene Yung, thank you for coming on Software Engineering Daily. It's been great talking to you.

[0:46:46.7] RY: Thanks for having me.

[END OF INTERVIEW]

[0:46:50.8] JM: GoCD is a continuous delivery tool created by ThoughtWorks. It's open source and free to use and GoCD has all the features you need for continuous delivery. Model your deployment pipelines without installing any plugins. Use the value stream map to visualize your end-to-end workflow. If you use Kubernetes, GoCD is a natural fit to add continuous delivery to your project.

With GoCD running on Kubernetes, you define your build workflow and let GoCD provision and scale your infrastructure on the fly. GoCD agents use Kubernetes to scale as needed. Check out gocd.org/sedaily and learn about how you can get started. GoCD was built with the learnings of the ThoughtWorks engineering team, who have talked about building the product in previous episodes of Software Engineering Daily, and it's great to see the continued progress on GoCD with the new Kubernetes integrations.

You can check it out for yourself at gocd.org/sedaily. Thank you so much to ThoughtWorks for being a long-time sponsor of Software Engineering Daily. We're proud to have ThoughtWorks and GoCD as sponsors of the show.

[END]