EPISODE 322

[INTRODUCTION]

[0:00:00.5] JM: Designers and software engineers need to communicate with each other. From Apple, to Slack, to Uber, the emphasis on visual design within a product is rising in importance. Much like development and operation silos have been bridged with the devops movement. Design and engineering teams are working more closely together than ever. They're trying to align the vision of the designers with the realities of code.

InVision is a product for prototyping designs and product workflows. I categorize InVision with other high level productivity tools, like Trello, Asana, and GitHub. When teams start using InVision, it often becomes an integral part of the entire product development workflow. A designer will mockup a product vision on InVision and share it with the rest of the team for criticism and commentary.

Since the product is a perfect bridge between engineers and designers, InVision, the company is expanding as a platform. Bjorn Freeman-Benson is the CTO at InVision, and his task is to scale the 100% remote engineering team while also developing new product features. Bjorn describes himself as a software psychologist. We talked about the psychology of managing engineers and the technical challenges of building a large popular web app for technology products.

I have used InVision personally. I use it to do design work for my company, Adforprize, and I vouch for it. This is not a sponsored post, but I love InVision. It was a pleasure to do an interview about it. I hope you enjoy it as well.

[SPONSOR MESSAGE]

[0:01:59.5] JM: You are building a data-intensive application. Maybe it involves data visualization, a recommendation engine, or multiple data sources. These applications often require data warehousing, glue code, lots of iteration, and lots of frustration. The Exaptive Studio is a rapid application development studio optimized for data projects. It minimizes the

code required to build data-rich web applications and maximizes your time spent on your expertise. Go to exaptive.com/sedaily to get a free account today.

The Exaptive Studio provides a visual environment for using back-end, algorithmic, and front-end component. Use the open source technologies you already use, but without having to modify the code, unless you want to, of course. Access a k-means clustering algorithm without knowing R, or use complex visualizations even if you don't know D3.

Spend your energy on the part that you know well and less time on the other stuff. Build faster and create better. Go to exaptive.com/sedaily for a free account. Thanks to Exaptive for being a new sponsor of Software Engineering Daily. It's a pleasure to have you onboard as a new sponsor.

[INTERVIEW]

[0:03:30.5] JM: Bjorn Freeman-Benson is the CTO of InVision. Bjorn, welcome to Software Engineering Daily.

[0:03:35.3] BFJ: Thank you. Glad to be here.

[0:03:37.4] JM: You described yourself as a software psychologist. What does that mean?

[0:03:42.9] BFJ: It's a term I made up, because when I sat down and thought about, "What is it I do as a CTO and a senior technical leader?" I realized I spend most of my time listening to either people, or listening to the code, tell me things about how it's in pain, or if there are people — How the people are in pain, then solving those problems for them.

I realized, "Hey, that's what a psychologist does. They listen to people lying on their couch who are in pain, emotional pain, or whatever, and then they help them solve those problems." I listen to people, and I listen to code, and then I solve those problems.

[0:04:17.1] JM: We had Yvette Pasqua, who is the CTO of Meetup, on the show recently. From what she said, it sounded like the CTO job is more about facilitating communications and

sympathizing the viewpoints of your best engineers, rather than breaking down the technical problems from a low level standpoint yourself. Does that resonate with your experience? Is that part of the software psychology that you're referring to?

[0:04:45.4] BFJ: Yeah. I think the CTO job varies in different companies. I think that what she and I are talking about is the role of a CTO isn't to do the project management and the detailed, say, function, specification of a particular piece of software. Rather, to give guidance and direction based on a long experience of what works and what doesn't work and help the smart engineers that we've got working with us to make the right decisions, the good decisions, that lead to really great products.

It's more a matter of setting the right context. In one way, in a previous like, I talked about making a wall at the garden. My job as a CTO is to define what those walls around the garden are to make them not too small so that engineers actually have an opportunity to make good decisions, but not too large, so that there's too many opportunities, make decisions, and you can make bad ones. If you define a nice walled garden, engineers can be incredibly productive inside that garden, and then we can have great products come out the other end.

[0:05:44.2] JM: We'll get into some of the decisions that you've made within the InVision product and within the team. Let's start by talking a little bit about InVision itself, because I've used the product for prototyping and collaborating. It was actually crucial for the early stage development of the product I've built most recently.

Explain what InVision's core product does.

[0:06:11.4] BFJ: What we're in the business of is digital product design, or actually, we're in the business of providing tools for digital product design. If you build a digital product, a web application, or a mobile application, or something along that line, you want to design it and make sure that you've got a really good product, not just technically behind the scene, but also from the user experience point of view.

Our product, the one that you're most familiar with, and then the follow on products that we have both out there today and are coming in the near future, expand upon that to make a digital

product design suite to make it really easy to build great digital products. To make all of the people involved in that ranging from that designers, to UX designers, to engineers, and even to executives who are watching the whole thing really productive and be able to create great digital products out there for their customers.

[0:07:04.3] JM: We've had a number of shows about devops, which is the movement of bringing development and operations closer together. I feel like similar silos exist between the product design side and the engineering side. Do you see those silos, and what are the problems that those kinds of silos can lead to?

[0:07:27.2] BFJ: Jeff, I got to say you're astute as you are handsome, because that is exactly the market that we're going after, is figuring out how to reduce those silos between a more traditional, let's say, artistic side of design, and the more traditional operational engineering side of engineering. Getting that whole digital product design experience from just thinking about the first idea all the way down to working code that works 100% uptime out on the system, that's the target market that we've got, and we are, as a company, aiming at it.

[0:08:01.4] JM: Let's zoom out a little bit. It's 2017 — Explain how design affects the engineering process at a typical company, because it wasn't always this way where design was such an important component of the engineering process.

[0:08:16.5] BFJ: I think the design has become increasingly important as the tools and technology that we use for building things on the engineering side have become more mature and more powerful. We've moved from the problem of just simply, "Can we do something?" which is what we use to do, say, in the previous century. It was just — We wanted to build a product just like, "Is it even possible to build something like that with the technology we have?" We've moved beyond that. We can basically, now, build anything.

Goodness gracious, we can build self-driving cars. We can build anything. Now, the question is not, "Can we build it?" but, "Can we build something that's a wonderful user experience that people want to use every day?" I think your smartphone is a perfect example of this. Can you imagine going back to the days of the Nokia brick phones and having something that you felt so

passionate about that you doodled on it and you had special colors and so on? Nobody did that for those phones back then. They were just barely functional.

Now, we have these beautiful phones with all sorts of powerful features, because the user experience is so much better, and that's because I think that the tools, maybe even the devops tools and the infrastructure behind the scenes has basically got to the point where it's become easy enough to actually build those things that now the distinguishing factor between whether a product is successful or not successful compared to another product, is in the design aspect.

Partly, of course, I'm going to say that, because I'm at a company like InVision, and we're passionate about design. Part of the reason I joined InVision is because I actually believe this and I actually believe this is where the future of product evolution is going to come from. Isn't so much from the operational side, the sort of CTO things I do, but from the design side that my colleagues do in a product design side.

[0:10:03.7] JM: The tools for managing and developing those products are getting better and better. First, we had things like Trello, and GitHub, then we got InVision. There's, obviously, Asana, and Slack more recently. There are so many other web-based tools that give people a lot of leverage. When these high leveraged tools come out, sometimes they improve the current workflow of an organization. Other times, they completely redefine how we do work. We've certainly seen that with Slack.

Now, these tools are integrating better and better with each other, it's almost like they're integrating as well as a software library that you just import. Are you seeing any interesting workflows developed from people who are using these high level tools together?

[0:10:53.4] BFJ: I am. As a company, we're seeing that on the design side. Design is a very — Donewell is a very collaborative process, and so you have to use lots of tools. There have been tools for decades that you can do design with, and people have lots of good tools they use and ways that they do that in workflow. You can see that integration and collaboration on the design side.

As a CTO, I'm looking at more on the technical side. Even there, we're seeing lots of integration between different things and some of the interesting workflows that we could do. I don't know how technical you want to get into in our discussion here, but we can go into some of the interesting technical ways that we're putting those workflows together to make our engineers substantially more effective, which, of course, then allows us to produce better products.

[0:11:42.9] JM: Yes, we will get to the technical side. Let's just start by talking about the core technology stack of InVision. Can you just describe the InVision tech stack, maybe the cloud provider you use, the basic workflow of your engineers. Give us a high level picture of the InVision tech stack.

[0:12:02.4] BFJ: Yeah. We're currently hosted on Amazon AWS. We run our infrastructure on top of that on Kubernetes and, therefore, Docker container is on top of Kubernetes, and that's how we distribute our workload across multiple machines. We use other Amazon, things we store our assets and S3. We've got an RDS instance, or 2, or 3, or 10, for storing the various pieces of metadata. We use SQS and SNS for queuing in between there. We're using a bit of lambda to do things like thumbnailing and image processing.

We're using that whole Amazon tech stack. Those Docker containers can be written in any language that the implementation inside can be in any language that the engineers find necessary to use. Most of them inside InVision are either written in Node, or in Go. Then, on the frontend, we have mostly React these days, although we still have some Angular from our previous evolution of the site. It's a fairly, I would say, classical tech stack there. The interesting thing comes in how we've put that together into our fully effective tool for building tools.

[0:13:09.3] JM: Absolutely, and you are describing a, like you said, fairly standard cutting edge stack. Although, I don't want to understate the fact that InVision has been around for a while, and I think it's oppressive that the stack has kept up with what is sort of contemporary — Over the last couple of years, we've seen this rapid adoption of Kubernetes. Before that, Docker, obviously. What was that migration process like? When did InVision decide, "Okay. We're going to move everything under Kubernetes. What was the process like?

[0:13:43.3] BFJ: Yeah. We started the migration from our original monolith, because like all startups, we started with a monolith. We started that migration, I guess, it's a little over two years ago now, breaking it down into a number of services and making that work. As part of that migration, we realized that we needed a better orchestration system than what we were currently using at the time.

A year and, probably — I guess, a year and a quarter ago, we started with Kubernetes back when it was fairly new, and we are actually one of the few companies who's actually running Kubernetes in production, and it's working really well, because it allows us to distribute the load across our machines in an interesting way.

One of the interesting ways that it allows us to distribute the load is we can actually standup private copies of our stack for important customers and without having to use additional machines, because we're having the same number of users that we would have supported without the private instances. Since they're distributed with Docker containers across the machines — We have the same number of users, so the same load across the machines. Now, we have these security isolated clusters for our larger customers.

[SPONSOR MESSAGE]

[0:15:03.5] JM: Good customer relationships define the success of your business. Zendesk helps you build better mobile apps and retain users. With Zendesk mobile SDKs, you can bring native in-app support to your app quickly and easily. If a user discovers a bug in your app, that user can view help content and start a conversation with your support team without leaving your app.

The conversations go into Zendesk and can automatically include information about the user's app information, device information, usage history, and more. Best of all, this is included with Zendesk for no extra charge. Use the out of the box iOS UI to get up and running quickly, or build your own UI and work with the SDK API providers. Keep your customers happy with Zendesk.

Software Engineering Daily listeners can use promo code sedaily for $177 off. Thanks to Zendesk for supporting Software Engineering Daily, and you can check out zendesk.com/sedaily to support Software Engineering Daily and get $177 off your Zendesk.

[INTERVIEW CONTINUED]

[0:16:31.5] JM: What are some of the subtle engineering complexities that you run into on a regular basis when you're building InVision?

[0:16:41.4] BFJ: The biggest problem that we have in sort of subtle engineering complexities is I would say one that every other company that's going through this transition of monolith to services encounters which is that a distributed system of services is very different in its debugging behavior than a monolith. It's much harder, for example, to put in breakpoints and to try out test data, because the things that might fail are not only the inputs that come in, but also the interaction between all those services and the underlying machines and databases that you have connected to those services.

It's been a learning experience for our engineering organization to get good at providing the right type of logging and monitoring and that sort of thing, so that we can actually find problems that either we've introduced or our customers have found in our systems.

That, I think, is the largest problem that you find when you're doing something like this. The actual breaking into separate services is exactly what you would expected to be. If you're a good software engineer, you're already modularizing your code. If you're just doing that in services instead of a monolith, that part works fine. It's just that the interactions, the interesting interactions between those servers is what is the interesting learning experience.

[0:18:03.4] JM: You must have had a front row seat to this growing importance of distributed tracing, and monitoring, and logging, because you were vice — I think — What? SVP in New Relic?

[0:18:18.5] BFJ: I was SVP of engineering at New Relic, from three engineers, to 300, as we grew that company. Yes. Actually, monitoring is one of those things I know something about.

[0:18:29.0] JM: Tell me a little bit more about what you learned at New Relic and how you have brought that to InVision.

[0:18:36.8] BFJ: One of the interesting things about the experience at New Relic is that the software industry was going through a particular transition in those years I was there, where it went from, really, monoliths were the way to build things. Of course, everybody built some services. Primarily, you built rather large pieces of software.

During that time, we moved from that rather large pieces of software model to rather small pieces of software. Now, we build large complex system out of lots of little pieces of software. Watching the change into the need for how you have to monitor, and do logging, and put in trace points and so on. Watching that transition as both New Relic and New Relic's customers went through that, has been really helpful as I've come to InVision, because, of course, InVision is going through that exact transition from a monolith, technically, into multiple services. Bringing that experience along and put it in the right pieces from the beginning, rather than having to go back and retrofit them.

[0:19:38.3] JM: There are so many people, as you said, who are going through this process. How does it affect how you prioritize different aspects of engineering and how you decide what the different engineering roles that you're going to have or going to be — Look, I just talked to somebody at Stripe yesterday who is head of the observability team, and he decided to make the observability team, because Stripe needed observability, and I was like, "That's interesting innovation. I haven't heard of an observability team before."

[0:20:11.2] BFJ: Yeah, that's great. We got to come up with better names for our teams too.

One of the things that I've done — Again, I'm not claiming to be unique here. All of my colleagues who have a similar CTO type roles are doing this same thing — Is divide the engineering team up into two pieces. There's the product engineering groups, and then there's the platform engineering groups. Each one of those has many teams, but I'm sort of in two large clusters there.

Then, platform engineering consists of building our specific platform on top of AWS that has the right abstractions for building digital product design tools, because our product organization builds products for digital product design. We want to use the right abstractions in there.

The abstractions that AWS provides for you, whether that's EC2 instances for computation, or serverless with lambda, or data storage, or whatever, they're great for generic computation, but when you talk about digital product design tools, or if you're talking about Stripe, they have a whole different product set. The abstractions you need for those are higher level than the straight computation ones.

For example, we have a lot of image uploads to InVision, because we have a lot of images. That's what people do with InVision. Part of our platform provides an abstraction for image uploading, and processing, and thumb-nailing, and resizing, and all these things, so that the individual product teams don't have to think about that at all.

Now, image uploading, you say, "Oh, well that's not that hard. I've got the internet, I'll Google a bit, I'll find something, I'll use that code." You could do that, but what I'm saying is we've provided something even better for our engineers, which is an existing service that already just works, and so you don't even have to think about it. Now, you can spend your time on the product features and not any of that image manipulation.

[0:21:59.2] JM: Incredible. What are some of the other things that you've build on top of Kubernetes, on top of AWS? What are the abstractions that you find your product design feature developers needing as building blocks for the products that InVision makes?

[0:22:21.2] BFJ: Yeah. Most of the abstractions that we have are the ones that you would commonly find in other teams as well. We have a feature flag system. We have permissions in users, in subscriptions, in accounts. We have authorization, authentication, all those sort of standard things. What's we've tried to do is we've tried to, in addition to those standard ones, find the other places in our product feature set that our engineer, our product engineers are spending a lot of time doing again and again and again, and then abstracting those out.

One of them that I talked about is images. Another one is synchronization. Today, the way people use InVision is they go to Sketch, or Photoshop, and they build their images and then they synchronize those over with InVision, and so that whole synchronization library back and forth is one of those abstractions that then once we've got that working as a platform feature, the product teams don't have to worry about that. They just use the data that's been synchronized.

The other thing that we've done with the abstractions is work on our deploy pipeline. In a minute or two, when you ask me the right question, I'll tell you about that deploy pipeline.

[0:23:29.2] JM: Okay. Well, I'll just ask that now. What do you do for continuous deployment?

[0:23:33.0] BFJ: Okay. One of the things that we've got is, I said, "One of our problems with building InVision is that the real world is a cruel, cruel place. Users can do things to your software that you can never imagine in a QA environment." You have QA people —

[0:23:51.2] JM: Then, InVision is very richly featured. For those who don't quite know what InVision, it is a very rich web-based platform. If you think about something that asymptotes towards the design richness of Photoshop, except it's in the browser, that's around where Envision sits.

[0:24:07.9] BFJ: Exactly. Our customers, our designers, people with a very high aesthetic sense, and so it's really important that our product matches that aesthetic sense and that usability. What we have discovered overtime is that no matter how well we build test, and integration test, and unit test, and how well our QA department works on the quality side, when we release something out into production, we find that the real world always finds some way to show us that we didn't test everything.

We use a fairly classic canary deploy mechanism, except that instead of randomly choosing the subset of customers who are going to get the new release, we've actually deliberately chosen a subset of customers by asking them. We've asked a set of our customers, "Hey, do you want to be on our early access program and essentially help us QA the latest releases that come out in return for seeing our latest releases?"

When we do our deploys, we deploy out to our early access canaries, which has about 10% of our production load on it. Then, we get real production load on our systems before they go out to the other 90% of our customers in a way that doesn't surprise anybody.

Then, once they've gone both through the early access load and worked great. Then, automatically, once they pass that to the production systems, and then they sit there for a bit, then they automatically go out to our private cloud customers who are our large customers who have private instances. They get the most well-baked version of the software that's deployed.

The way I'm describing it here, this sounds like a slow process. It's true, the latency of any given deployment can take two to three day to get from deploy out to the private cloud customers. At the same time, we're doing 40 to 50 deploys a day. There's a queue of these things all working their way through this deploy pipeline, which has a two to three day latency.

It's really slick. We do these all with Kubernetes. Each new deployment deploys out a new set of odds running on the Kubernetes thing there, and it just automatically switches over to the new one and it sequences each new set out to the correct set of machines and customers at the right time. The engineers just have to say, "Deploy to our chat bot," and it deploys their code out the chat bot, and then the rest of it happens automatically behind the scenes.

[0:26:36.4] JM: That sounds great, except that if the engineering team grows at a rate that outpaces your ability to canary these things, then you could see that feature queue getting longer and longer. Are there opportunities that emerge different — Yeah, go ahead. How do you deal with that?

[0:26:53.7] BFJ: Exactly. The fact is there are multiple canaries going on at the same time, and so that's how we handle that so that we can do parallel deployments, essentially. If you deployed at 10 a.m. and I deployed at 11 a.m., your canary test is still going on, because maybe you're going to test for 24 hours before you promote yours.

Since we're a SAS company that deploys in an almost continuous delivery mechanism, we can't wait for you to finish for me to do mine, and so we just deploy them the same way we do today.

We just deploy them over each other, and there are multiple ones going on at the same time. It's actually working extremely well.

[0:27:33.3] JM: Sure! Of course. I worked at Amazon briefly, and I witnessed a similar system inside there. I'm sure every company has similar systems with overlapping tests. Maybe, sometimes, you have some disjoint tests if you have a really big feature that you want to test in isolation. I'm curious about build versus buy, because you're talking about a lot of things that — It sounds like you have to build these abstractions for product design — Designer products, designers, or whatever the abstractions that you're building for, people who are working at InVision, and then you've also got a lot of buy decisions. You talked about RDS. You're talking about continuous deployment. I don't know if you use CircleCI, or if you use something in-house. Talk about how you resolve questions at build versus buy?

[0:28:19.3] BFJ: Yeah. I would say that as a good startup CTO, you should never build anything if you can buy it, because everything at this stage of a company's life is about velocity. Being able to try experiments faster than you ever thought possible, and hopefully faster than any of your competitors.

You can really only do that by using — Let's call it off the shelf, although now it's internet SAS software — Off the shelf software as much as you can. By enlarge, I try and use things out there that I find out there rather than building our own. For instance, we use existing logging systems rather than building our home. We use an existing feature flag system instead of building our own, and so on.

We only put in our own abstractions where we need to do that where we can't find an existing product, which is good enough for what we need. Of course, good enough — No product is going to be exactly what you want, so you have to adjust your expectations a little bit to match what's out there. For the most part, we can find a lot of those things, but, again, not all of them, because we're in a particularly unique product space, and so nobody's going to build every single abstraction that we need.

[0:29:36.4] JM: Given that InVision is a product that is focused on design, and you have many features that get designed. I am curious about the workflow for feature development at InVision

itself. How the product designers interact with the engineers and how you shift things out? I guess this would more be on the product engineer side as supposed to the platform engineering side.

[0:30:03.8] BFJ: correct. You can imagine that — Of course, we use our own product, because that would be sort of silly if we didn't, since that's what we do. It's a very iterative process that the engineers and the designers are working together in a very quick iterative mechanism so that we can get things out and try experiments as quickly as possible.

I have insisted that all the engineering teams delivery something useful, at least once a week, to their product managers. We have weekly demos of all the new stuff that comes out, and that forces a particular cadence, which forces a particular type of interaction with the product designers to allow rapid iteration on possible concepts and the way the flow works and so on.

This only can happen if you have an underlying system that allows you to code, and deploy, and experiment extremely quickly. I think that the magic sauce in anyone of the startups, like ourselves, or other similar startups, is that ability to iterate very quickly. I think that's it's really part of the CTOs role to build an infrastructure that allows the company as a whole to do those rapid iterations.

[0:31:18.8] JM: You strike me as somebody who has a disciplined approach to product development, perhaps engineering management. You said you have this rule where you have one, somebody — That your engineers have to deliver one thing of value per week. Can you talk more abstractly about your philosophies around engineering management and how you set that product cadence?

[0:31:42.1] BFJ: Accept that I'm going to correct you and say at least one thing of value. I'm thinking I could do more than one thing of value.

[0:31:47.4] JM: Okay. Right.

[0:31:49.1] BFJ: Yeah. My philosophy around management is that the goal is the management should set those walled garden edges around and hire great people, and then in some sense,

step back and let them do fabulous work. I understand that whenever I tell people this, they go, “Well, of course, a manager should do that. Every says a manager should do that.” My manager was a micromanager, and he is like —

Yes, what I’m saying is this thing that everybody says when they talk about management. It’s like, “Oh, I’m an enabling manager.” That’s what people say. In my case, I actually practice that to the point where I try not to even show up on stage often. I try and have the engineers who did the work get up and talk about it. Perhaps, someone to my detriment as a CTO is that I don’t have a public enough face, because I’m always promoting the engineers, like, “You should get up there and talk about that,” and so on.

It’s really my model that the way to succeed in this business is to hire fabulously awesome engineers and then just set them up for success. Part of the ways that I’ve set them up for success is, over the years, I’ve — Let’s say, back when I was a coding engineer, I had all these things that I kept having to do that I hated doing. Total waste of time. Meetings, is your classic one.

When I was an engineer, a coding engineer, like, “Oh! I hate meetings. I can’t stand it. I don’t understand why they have us go to all these meetings?” I said, “If I ever get to be in charge, I’m not going to have many meetings.” One of my philosophies is engineers shouldn’t go to a lot of meetings.

Pair programming, I don’t count that as a meeting. That sort of thing, I don’t count as a meeting. Meetings are sort of the classic Dilbert-like thing, and engineers shouldn’t have to go to those, because engineers should be creating that product that create digital product design tool that we’re working on.

My management philosophy is around a lot of these things, it’s how do you set up an environment that’s fabulous for being an engineer? What are all the things as a software engineer that you want to do, and how do I set up an environment that just makes it the easiest path be those things?

[0:33:56.4] JM: What would you do when somebody's applying to InVision and they exhibit characteristics of excellent product development, excellent engineering, but it seems like they need a lot of management. Maybe they've been conditioned, like all of their previous managers have been micromanagers. Is that ever a point of friction in your management style with people you hire?

[0:34:22.8] BFJ: I think it would be both a friction with my management stuff, but it also just doesn't work at a distributed company like InVision. InVision is actually unique — Or maybe not unique. It's interesting in the sense that we don't have any offices. We all work from home. All 200 plus of us work from home, or coffee shops, or wherever. Because of that, you can't hire people who require supervision. I'm calling it supervision, rather than management. Somebody who watches over you and make sure you work. You have to be a self-motivated driven person to be the sort of person who works in an environment where your boss can't actually see you at all. The only thing we can measure in the engineering organization is did you produce results?

If you happen to be the world's most brilliant programmer and you can do everything that you've agreed to do in two hours per week, I would have no way of knowing, except that'd I'd be completely happy, because you'd be producing all sorts of great results. That particularly problem is people who need a lot of supervision, just wouldn't work at distributed company.

[0:35:25.3] JM: For people who are refugees from that type of environment though, because I know people who have only worked at those types of environments where they're used to being handed tasks, and they do those tasks. It's like, "You don't have to work that way, and you'd probably work more productively if you weren't working that way," but they've sort of been North Korean into this way of working. How do you — Maybe you're not talking to somebody, talking to an engineering who's at InVision. You're not talking about them from a management capacity. You're talking to a friend and you're trying to say, "You know, if you're working this way, it's a sign that you're in the wrong sort of organization." How do you convey that to people?

[0:36:07.6] BFJ: Pretty much those words. I think I'll just take the recording of this and play it back to them. I actually might disagree with you slightly there. I think that there are a lot of different company cultures that one can imagine and, actually, it exist, and different company cultures work for different people. There are different times of people's lives, different things that

are important to them that would cause you to choose one type of culture over another type of culture.

I have friends still at Microsoft who've been there for a long time. They love that place. I probably would not be successful at Microsoft. That particular culture doesn't match with my vision of the way software should be done, but friends of mine love it. Similarly, at Amazon, I have friends at Amazon, and they're very happy working there. I probably would not be a good choice to be somebody working at Amazon.

What I'm saying is I don't think there's one way that things need to be done, and so if you're one of those people who likes a particular culture, even if, I, Bjorn, don't like that culture, it doesn't mean that you should necessarily leave it. I think that the right thing to do given that there's just such a wide variety of companies out there doing such amazing things of all types, is to find one that matches your particular culture.

I think as you find more companies that do like InVision and allow people to work anywhere in the world, you're not as constrained to what's available in your geographic area. My best programmer lives in Montana. It's fantastic. He might be the only programmer in Montana for all I know, but he didn't have to move to the Bay Area to be a star programmer in startup.

[SPONSOR MESSAGE]

[0:37:57.2] JM: Release the Kraken! GitKraken, that is. Are you tired of feeling like you're sailing the stormy seas, because you have a clunky, old Git user interface? Unleash the beast, that is Axosoft's GitKraken. Voted 2017s most popular Git GUI for Windows, Mac, and Linux.

GitKraken is designed to make you a more productive Git user. The app offers efficiency, elegance, and reliability. The UI equips you with a visual understanding of your branching, merging, and commit history, and features multiple profile support, one click undo and redo, a built-in merge tool, and fast search.

Run the installer, open the app, and set sail with GitKraken. Easily setup integrations with GitHub, GitHub Enterprise, Bitbucket, and GitLab. That's one high performance sea monster.

Visualize your version control and code on into the sunset, sailors. Visit gitkraken.com/sedaily and use promo code “sedaily" to get $10 off GitKraken Pro.

Thanks to GitKraken for being a new sponsor of Software Engineering Daily.

[INTERVIEW CONTINUED]

[0:39:23.5] JM: For sure, and I really like what you say. I think it sounds like I need to work on some of my acceptance. As you’ve said, InVision is an entirely remote company. This, to me, is a bellwether of where we’re going. There’s a lot of debate around the viability of remote companies. I remember when Sam Altman who is in charge of Y Combinator, when he was talking about, “Oh, if you have a remote company, it’s an AntiPattern. It’s obviously a sign that this is a poor investment.”

This is a guy that’s on the cutting edge, and I think he was saying this two years ago. He might still be saying it, but we have moved to a place where remote companies can be tremendously successful. In some ways, it’s so advantageous to having a remote team in contrast to an on-prem team, or whatever you want to call it. Let’s talk about some of those costs and benefits.

[0:40:16.5] BFJ: Yeah. I think Sam has got a good point, which is that it’s actually harder to build a remote company, because you need to have a better management team to do that. Because, for example, you can’t see what people are doing, so you have to be better at being able to specify and watch through remote indications, rather than actual physical indications.

When you have a young founding team who has, realistically, no management experience, it is an AntiPattern for that to be a remote team of any size. Maybe it’s three people who all work together when they were in college, and so they know each other, and then they’ve moved to separate cities. Susan is in San Francisco, and Jane is in Seattle, and Bob is in North Carolina. That might work, but when it gets any larger, they don’t have any experience at managing that, and it sorts of falls apart.

I think Sam has a good point there. Having said that, if you put it in place, a management structure of people who understand what needs to be done and have seen a little bit more of the world, you can actually build an incredibly effective remote organization, because you can take advantage of talent wherever it is in the world and you can basically allow people to live lives that doesn't consistent entirely of sitting on BART, commuting back and forth, and so then they can apply more of their mental energy into building, in our case, great digital product design tools.

[0:41:43.9] JM: Are there certain types of products that are better suited to having a remote engineering organization?

[0:41:50.4] BFJ: Gosh! In my opinion, remote organizations only work if you can break the work up into small enough pieces. If you need an engineering memorization of 50 or more engineers, or even 20 or more engineers to do some particular project, that's not going to work very well in a remote setting. That's my opinion.

One way we set up the InVision engineering organization is we have lots of little small teams. Feature teams are three to five engineers, and that three to five engineers, the Slack channel isn't too noisy to distract me from doing my work. We could even open a Google Hangout all day long and just sort of hangout on that thing and chat back and forth to each other.

Beyond three, maybe four people, that's going to start to become distracting and you're not going to get any work done. You don't want to do that with a larger team. If you can build your product in a way where you have lots of small features and they can collectively work together to become a coherent whole for your customers, you can then build, I think, a very effective large distributed organization out of lots of pieces. Sort of like Lego bricks.

[0:42:55.9] JM: It's an interesting tension. When I was at Amazon — Obviously, Amazon is the namesake of the two pizza teams, and yet Amazon buildings compose much of downtown Seattle. It is, in some sense, monolithic, because all of these buildings are in close proximity. Amazon is tightly coupled with the rest of the city. It's just a giant organization, and yet the company manages to get broken down into these two pizza teams that are in some sense decentralized. Maybe, you could argue that the communication frequency among different

teams is noisy enough to endorse a monolithic MeetSpace architecture. It's interesting tradeoff. How did you see — Doesn't New Relic have some remote people? How did you see that playing out at New Relic?

[0:43:53.0] BFJ: At New Relic, we really didn't have very many remote people. We had a few, but primarily, it was all collocated in one place.

Yeah, the issue of Amazon is really interesting, because, of course, both in practice and various studies that have studied this, show that once you have an organization that has multiple floors in a building, you might as well be on different planets. It's really an interesting thing.

Given that Amazon is that large, I don't see why they don't distribute themselves out to more cities. They would be just as effective, because the people in building one don't see the people in building to any more than if the people in building one were in Seattle and the people in building two were in Denver. It's really hard to argue with their success, because Amazon has been fabulously successful. There is probably something there that I don't fully understand, that they need to have that proximity about.

[0:44:50.6] JM: They can calibrate the Kool-Aid in the pipes a little bit more easily if they are all in one place.

[0:44:58.2] BFJ: Counter argument. Google, has labs all over the place, and they're also very successful. The Google model is not that they're completely distributed, they just have many, many labs. I think perhaps they just have better Kool-Aid instrumentation in their pipes.

[0:45:14.0] JM: Actually, I would argue the contrary. I think Amazon is much stronger. Look, this is very speculative, but the gravity of Amazon is pretty tremendous. There're so many people who leave Amazon and they'll go to a place where it's a startup with more velocity, or they go to Google and there's infinitely more perks, and they have a higher salary. Yet, they return to Amazon. It's a very interesting phenomenon.

[0:45:41.5] BFJ: Yes. This is back to my previous point that there are company cultures for every personality. I think that when you find one that matches the way you like to work and the things that motivate you and drive you and excite you, then that's where you stay.

There are plenty of people who go to work for Amazon and are very happy there and stay there for a long time. Like I said, I know some of them, so they're not even abstracts, they're not even abstractions, they're real people. Yes, I can easily see that if you've enjoyed that particular match, you go somewhere else and discover that it's not the right cultural match for you. That you would go back to the original cultural match and have a great time there.

For example, here at InVision, a number of people have followed me from New Relic, because they liked the world that I build for engineers, and I'm building pretty much the same cultural world that I did before. It's the thing I do. After I left New Relic, the change the culture there, and they change to its culture that those people didn't particularly like. They said, "Gosh! I really like that culture." "Look, there's another one over there. I'm going to go join that one." I can easily see people returning to Amazon, and I even see that as a good thing.

[0:46:54.4] JM: You mentioned Slack. I think of Slack as really crucial to this remote work phenomenon just becoming so much more smooth. Obviously, there was a surface area of companies that could do remote engineering before Slack. I think that service area has increased with Slack. Can you describe how you use Slack and why it's so fundamentally useful?

[0:47:19.9] BFJ: I think that the key to making remote work work is to have the right set of communication paths through all the different ways that you need to communicate. Slack is one of those communications paths. Google Hangouts, or in our case, we use Zoom, is another one. Email, as primitive and old fashion as it is, is yet another one. You have different mechanisms for different types of conversations.

Prior to Slack existing, there was a real hole in that sort of almost real time, but still a synchronous chat style communication. People used things like Instant Messenger, and so on, but it wasn't quite the right solution for that problem. When Slack has come along, it's really solved that particular hole. We're, again, fairly classic. Like most companies, we have individual

channels for teams. We have individual channels for topics. We have larger channels for the whole company, and larger channels for all of engineering. Individual chats back and forth with people during the day.

It's sort of the equivalent, I think — Some say that it's better than if you're all in one location. You might walk by somebody's desk and leave a post-it note saying, "Hey, when you get a chance, give me a call. I want to chat about the architecture of our new database." You just leave a note for them. Then, they'd come over and chat with you. That's sort of what you can do with Slack now, but you can do it in a distributed sense.

[0:48:52.7] JM: What are the opportunities for machine learning in InVision's product?

[0:48:57.0] BFJ: At the moment, we're not doing much with machine learning, because InVision is really about the creative side for digital product design. We don't want to build a tool that attempts to take away that creative aspect. There are things that, in the past, you might have called machine learning, like gesture recognition and so on, but I wouldn't call that machine learning anymore. Machine learning would be taking large datasets and trying to find patterns in there that you could automate.

Since we're not trying to automate the digital product design experience, we're trying to enable these great designers that we find out there as our customers to do digital product design. We're not actually spending much time on machine learning today.

[0:49:36.5] JM: Can you imagine a place where you would get there? I totally sympathize with your point. I can also imagine a future where machine learning become — The tools that we have today will just seem so overly complex and hard to use. It'd be like comparing building a microservices architecture in Java Enterprise 15 years ago, to what you do today to build a microservices architecture. Are there abstract opportunities that you can imagine?

[0:50:08.0] BFJ: I can imagine that we'll do some work on our underlying platform to use some of the large amounts of data that we produce in an operating system to predict where we're going to have failures and problems so that we can do a better job of running that service under the covers. What I don't see us doing — Again, I'm the CTO. I'm an engineer. I'm not the

product guy. What I don't see us doing is building sort of big data machine learning things that try and do the designer's job, because we really want a creative aspect of building a great user experience to be that human creative aspect.

I suppose one could build something that say, "Looked at all the products that —" Let's say you had a big customers, let's say Costco. Costco builds a bunch of things, digital product design — Digital products. Then, we could sort of look at all the way those were built and then predict what they're going to build next. In some sense, we could do that. You could imagine a system that would do that.

I remember decades ago, there were AI-like systems that could take, say, all of Beethoven's symphonies and then automatically generate a new Beethoven's symphony that had never been written before. You could imagine something like that. I just don't see us going down that as a product path anytime soon.

[0:51:26.4] JM: To close off, I know we're up against time, you mentioned AWS Lambda earlier. This is part of this serverless movement. The serverless cadre of products can, in its broadest sense, include stuff like cloud vision, the cloud vision — Google's image recognition API, or maybe things like Twilio. Then, obviously, at the most precise narrow definition, you have just stuff like AWS Lambda, or Google cloud functions that are super cheap computation that you don't really have a server associated with it, which is why it's so cheap. What are the opportunities for those type of products that you see for InVision?

[0:52:06.1] BFJ: Yeah. It turns out that the advantage of lambda or any of these serveless things isn't the fact that it's cheap. In fact, I bet you, if you did the math, it would come out to be more expensive than if you were able to do the same amount of thing on your dedicated hardware.

The advantage is you don't have to pre-predict load, and that's the fabulous thing, because what it's doing is it's back into my model of abstracting out the things that are not essential for the developer's to build the product. They no longer have to predict what the load of their particular system is going to be. Then, we can accommodate any load that our customers throw at us.

That removes that concern from the developer's plate, which then allows the developers to spend their time thinking about other tools that the digital product designer will need, and that's where they could spend their brain power. For me, it's not about the saving and the money. In fact, having done the math, but I'm pretty sure it's not really a savings of money, but it's about the flexibility and the removal of yet one more concern from what the developers have to think about.

[0:53:08.1] JM: Bjorn, thank you so much for coming on Software Engineering Daily.

[0:53:10.5] BFJ: I appreciate the opportunity. It's been a great chat.

[END OF INTERVIEW]

[0:53:17.5] JM: A few quick announcements before we go. Software Engineering Daily is conducting our annual listener's survey, which is available on softwareengineeringdaily.com. You can click on the survey link. The survey really helps us understand our listeners and gives us data that we can show to advertisers that help get us better sponsorship deals.

Also, the Software Engineering Daily community has started working on Mineranker. This is an open source newsfeed platform. We are trying to democratize the idea of a newsfeed so that the only newsfeeds in town are not necessarily Twitter, or Facebook, or any other centralized newsfeed. We'd like to make it possible for anybody to make a newsfeed.

You can check out the Mineranker Project at mineranker.com. You can check out and implementation of Mineranker at softwaredaily.com. You can find links to all of these stuff at softwareengineeringdaily.com. There you can also find a link to join our Slack Group, to follow us on Meet Up for future meet ups, and other information.

Thanks again for listening.

[END]

