EPISODE 834

[INTRODUCTION]

[0:00:00.3] JM: Alex Balazs is the Intuit Chief Architect and has been working at the company for almost 20 years. Intuit's products include QuickBooks, TurboTax and Mint. These applications are used to file taxes, manage business invoices, conduct personal accounting and other critical aspects of a user's financial life.

Because the applications are for managing money for users, there is not much room for error. When Intuit was started, the company made desktop software. In his time at Intuit, Alex played a key role in rearchitecting the monolithic desktop applications to be resilient, reliable web applications. Intuit originally managed the software on their own servers. Since then, Intuit has migrated to the cloud using AWS.

Alex joins the show to discuss his experience scaling Intuit, his strategy for cloud migration and his evaluation for criteria of questions of build, versus buy. It was a great episode covering a whole lot of history, as well as modern software architecture. Hopefully, we can have Alex back on the show at some point to discuss some of these subjects in more detail.

The FindCollabs Open has started. This is our second FindCollabs hackathon. You can find out about it by going to findcollabs.com/open. The prizes at FindCollabs Open our $250 category prizes for things, such as the best machine learning project, the best ReactJS project, the best cryptocurrency project, the best computer game, or game design. You can find all those details at findcollabs.com/open.

FindCollabs is the company I'm building. It's a place to find collaborators and build projects. I would love to see you in the FindCollabs Open. If you have any suggestions or feedback on the product, you can e-mail me, jeff@softwareengineeringdaily.com.

[SPONSOR MESSAGE]

[0:02:11.3] JM: Azure Container Service simplifies the deployment, management and operations of Kubernetes. Eliminate the complicated planning and deployment of fully orchestrated containerized applications with Kubernetes.

You can quickly provision clusters to be up and running in no time, while simplifying your monitoring and cluster management through auto upgrades and a built-in operations console. Avoid being locked-in to any one vendor or resource. You can continue to work with the tools that you already know, so just helm and move applications to any Kubernetes deployment.

Integrate with your choice of container registry, including Azure container registry. Also, quickly and efficiently scale to maximize your resource utilization without having to take your applications offline. Isolate your application from infrastructure failures and transparently scale the underlying infrastructure to meet growing demands, all while increasing the security, reliability and availability of critical business workloads with Azure.

To learn more about Azure Container Service and other Azure services, as well as receive a free e-book by Brendan Burns, go to aka.ms/sedaily. Brendan Burns is the creator of Kubernetes and his e-book is about some of the distributed systems design lessons that he has learned building Kubernetes.

That e-book is available at aka.ms/sedaily.

[INTERVIEW]

[0:03:46.8] JM: Alex Balazs, you are the Chief Architect at Intuit. Welcome to Software Engineering Daily.

[0:03:51.5] AB: Thank you, Jeff. It's great to be here.

[0:03:54.0] JM: I'm looking forward to talking through some of Intuit’s architecture. You've been there for, I think 20 years. Is that right?

[0:04:00.7] AB: Yeah, almost 20 years, since 1999.

[0:04:04.1] JM: Okay. That is a long time. I'd like to start with some high-level discussion of Intuit's product line, then we'll get into the engineering. Describe the different products that you help with the architecture of today.

[0:04:18.3] AB: Yeah, absolutely. Intuit's mission, which was recently refreshed is to power prosperity around the world. One of the cool things about that mission is that it's pretty open. Prosperity isn't necessarily something that you can measure concretely. It's something more about how people feel. Do they feel prosperous?

Our flagship products of TurboTax, QuickBooks and Mint are certainly a great start. They are financial products that help our customers, whether those customers are consumers, self-employed or small business owners, to feel more prosperous. One of the really cool things that's going on at Intuit is that we're growing out of our roots as a product company. Still remaining customer obsessed, but really becoming a platform and data company.

As you think about Intuit's product line going forward, certainly we will continue to maintain the amazing products we already have, but there's a lot planned in terms of Intuit becoming this platform company.

[0:05:15.8] JM: You've been there for a long time. What are the canonical problems in the Intuit engineering stack that were present in the early days that still exists today?

[0:05:28.1] AB: Yeah. It's a great question. The main problem that Intuit has been approaching is around finances. It started in the early 1980s when Scott Cook founded Intuit when he saw his wife struggling to balance the checkbook. That represented the first, what we call a follow me home. Scott pioneered the idea of the follow me home, which basically I know it sounds really creepy, but basically, what it means is we want to see how people experience finances and money in their own environment, not inside of our labs.

Obviously, we have usability labs where we test some of our new software, or we do a lot of it obviously online in terms of beta testing. One of the best ways to really understand your customers, to actually go to their environment and see what they do in their environment. The canonical problems that we've been solving is really helping our customers understand their financial situation. It's such an amazingly nuanced thing. It's not a one-size-fits-all approach.

To some degree, it's almost amazing that we were able to create software. Scott in the early days was able to create software in the 80s based in DOS, then through the 90s Windows. Then in 1999 when I was hired, I was part of the first team to actually put QuickBooks online. When I joined in 99, QuickBooks was a 100% desktop, TurboTax was about 99% desktop.

The problems that we're looking to solve is these nuance problems around finances, so how much money do I have coming in? Where am I spending my money? How do I predict the future? The great opportunity for us is that as much as we were in a position to solve these problems 10, 15 years ago, in the age of AI, in the age of predictive analytics and machine learning, we can do an even better job of predicting people's finances and helping them. It's the same canonical problem that we were solving 20, 25 years ago.

[0:07:29.9] JM: This issue of moving a client application onto the cloud is something that I've talked to few people about. Basically, many different companies have dealt with this. I think Microsoft dealt with it with visual code, or visual code, wherever their IDE is. I think that thing is in the cloud now, or Excel. I think Excel is in the cloud now.

[0:07:55.1] AB: Yeah, Adobe is a great example; open Adobe tools. Yeah.

[0:07:57.9] JM: Totally. Photoshop. How hard is that to do? Or what do you have to do? How do you go from a monolithic server – or client-side thing, to breaking up this monolith and moving it into the cloud?

[0:08:13.7] AB: Yeah, so unfortunately, the first step that you almost always take in going from one monolithic desktop app is to get to a monolithic web app. Some of the early generations of the products that Intuit developed online in the late 90s and early 2000s were pretty monolithic. They were pretty much direct representations of the desktop products.

The challenge that comes is now working in a distributed world. There's so much power and value that you get in terms of consistency of how you build when everything is enforced by the linker and the compiler. Now suddenly, you work in a distributed world where you have microservices and you have fine-grained experiences and you have widgets and you have composite applications and all of that. Managing that complexity while still enabling your organization to move fast is the biggest challenge.

I would say that doing software architecture in a desktop world is something that you can maybe do once and then you just reap the fruits of the labor. Distributed architecture in a SaaS world, in a platform world is vastly different, because it requires you to be really agile about how you think about architecture. The same way that software development itself is agile, you have to be agile about architecture and adapt and grow as the software changes.

We are constantly in the business of thinking about our architectural decomposition as business capabilities. What are the business capabilities that we're exposing? In the world of desktop, it was simple. You had this product and then the product had features, then you figured out it wasn't you released it. In the world of online delivering these products, it's more about what business capabilities do you want to expose? Then what's the technical architecture to deliver those business capabilities? It's a challenge, but it's fun. I mean, I think it's one of the funnest parts of the job.

[0:10:09.4] JM: I did an interview recently with the Airtable CEO. Airtable, have you heard of that company at all? If you looked at –

[0:10:18.1] AB: No. Unfortunately, I haven't. No.

[0:10:19.2] JM: Okay. Airtable is a modern spreadsheet/low-code system. They're pretty popular. In order to get the performance that they wanted out of this basically, a web spreadsheet, and in order to compete, they are a serious viable competitor to Google Sheets, which is incredible. The way that they got there was they actually had to build their own JavaScript framework, they had to build their own database. They literally built from scratch a database to handle a spreadsheet.

It just made me really think how performant are the web applications that we're using today, because a spreadsheet is the iconic, dynamic web application. It just makes me think like, how much more performant can things get? Maybe we're just in this infancy, where things are going to feel so much faster in the future.

[0:11:19.1] AB: Yeah, absolutely. We are constantly on a journey to try to figure out what are the parts of the stack that are the most critical for us to own, versus use from somewhere else, to actually deliver the experience to our customers that they want? When you do that analysis, there's a functional view that says well, we're in the business of finances, so we have an accounting engine and we have a personal finance engine. Then there's an element of it in terms of the implicit requirements that our customers have, in terms of the quality attributes of the system. Certainly, performance is one of them.

The problem with frameworks is that they're frameworks and they're designed for many. If that's not the differentiating part of your stack, frameworks work really well. If it is the differentiating part of your stack, then you may have to actually build something on your own, because that in fact is one of the most important things that you do. For us for example, in the world of TurboTax, TurboTax is basically a big declarative system. It encodes the knowledge of the US tax system in a big declarative system.

You can think about it and say, "Well, that's just a rules engine, so let's go take a rules engine off the shelf and just encode the rules." In order to get to the performance that we want, the speed that our customers expect, we actually couldn't do that. We ran certain scenarios, we ran certain tests that didn't work out, and so we decided that one of the core things to our businesses, in fact, the way that we build this tax engine that allows us to encode the tax law as content.

I think what you'll see is that the performance of online TurboTax every year gets better and better, as the product actually becomes more and more complicated in terms of the personalization.

[0:13:08.8] JM: Fascinating. By doing that, basically declarative syntax for describing how TurboTax works, I can see that even having additional benefits, because I was wondering about the fact that you need such domain expertise for computerizing taxes, I can imagine adding an additional interpretability layer, or perhaps – that would give you a great space to collaborate between domain-specific experts and engineers.

[0:13:43.0] AB: Yeah, absolutely. In fact, that's the key. If you think back about the early days of the internet, everything was in code, right? The user experience was in code. Maybe everything except the data was in the code. Then what you do is you try to – you start to say, “Well, if I want to deliver this amazing experience to my customers, what experts do I need?” The first kinds of experts that we exposed in our systems were designers. We said, “Well, what if we separated the way that designers create content and actually meld that together with the software?” Okay, great.

Then as you think about these software systems becoming more and more intelligent, more and more life-like and the ability to encode different types of knowledge, then you say well, there's other kinds of experts. Some of these experts are tax experts. One of the things that we take pride in is our ability to create the right tooling and engines that actually allow us to work with these domain experts, to actually encode their knowledge into the software.

[0:14:44.1] JM: Many companies that have been around for 20 plus years, or around 20 years, or even shorter time, 10 years, they're either making the transition to becoming a software company, or they have been a software company. There's a smaller subset of those companies that have been a software company the whole time. Intuit is a software company, but you cross the chasm into the post-cloud world. How do you use cloud providers, or how do you think about your consumption, or your strategy around cloud?

[0:15:22.3] AB: Yeah, it's a great question. In terms of before, what I was talking about this evaluation that we do, that we call core versus context. What is core to the business? What is differentiating? Why are we going to win? To us, what is context? Therefore, what do we want to take advantage of some of the best builders out there?

Cloud is definitely one of those huge enablers. Intuit, like many other companies 15 years ago was in the data center business. We had our own data centers. Most of these in fact, we just very recently are starting to shut down. In terms of our –

[0:15:57.6] JM: Kudos. Kudos by the way. That's tough.

[0:16:00.4] AB: Yeah, it is tough. It is tough. We developed a great strategic partnership with AWS. Amazon has been an amazing partner. Over the course of the – our cloud journey started about seven, eight years ago, went into overdrive probably about three years ago. Now every major Intuit product is running on the public cloud, including all of the data. It's been an amazing partnership, an amazing journey.

For us, the ability to scale up and scale down; you can imagine that TurboTax is probably one of the pickiest businesses that exists, other than maybe Black Friday for some of the e-commerce retailers. For us April 15th, which we just hit April 15th, to sit there and look at the dials, to look at the load coming in when you own the hardware and you know what your maximum capacity is. Basically, you have a bunch of contingencies in place. Like okay, what contingencies am I going to pull should something happen? Or what contingencies do I pull during our Super Bowl commercial, right?

In the world of the public cloud, you just scale up, right? Then when you're done, you scale down. Certainly, infrastructure as a service has been a huge win for us. Then now, what we're really doing is diving even deeper into platform as a service, so some of the managed services in AWS, and continuing to partner with them to help us with these managed services. Really, the way we look at it – it's funny to think about it that way, because Amazon is much more of a bigger company than Intuit, but we almost view them as an outsourced partner of Intuit, right? Because the partnership is so deep and there's such a – they're customer obsessed the way that we're customer obsessed. They're a great engineering company the way that we're a great engineering company. It's fun to partner with them.

[0:17:50.2] JM: Does that hard dependency scare you at all?

[0:17:53.3] AB: In terms of being dependent on AWS, not really, not really. I think that you always get something and you give something, right? For us, we believe that the partnership that we have with Amazon will allow us to move faster and to deliver more. Then the truth of the matter is that technology moves so fast, and your need to refactor and redevelop code and rewrite code is growing, right?

The days of okay, I'm going to ship something and then I'll plan to rewrite at 10 years from now, or something like that. You just can't do that anymore, right? You're constantly in the mode of saying, "How do I do this better? How do I do this faster? How do I rewrite this? How do I decompose this in even simpler ways?" In that world, we're constantly looking to refresh our stack anyway. Whether that refresh continues use to happen in AWS, or someday in the distant future it's not AWS, it really doesn't scare us, because that's the game that we're in.

[0:18:56.1] JM: Yeah. I mean, this whole multi-cloud conversation, I don't know how to feel about it, because we only have one JavaScript package manager, right? We have NPM, that's it. Yeah, a vulnerability can make it into the NPM supply chain. Vulnerabilities make it into hardware supply chain. We can have vulnerabilities exist in package managers. We can have zero days in our operating systems. I mean, you always have to ask this question, like what are the places where I'm going to put redundancy in? I don't know if this the whole multi-cloud thing is one of these things that we want to worry about. I'm honestly undecided.

[0:19:37.1] AB: Yeah. At some point, you have to trust something, right?

[0:19:41.2] JM: Right. Totally. The power grid.

[0:19:43.3] AB: Yeah. We have to make those choices. Whether those choices are ones on security and what do we do about it, what are our contingencies, whether its availability, we are constantly wargaming, we're constantly planning, we're constantly looking at the contrarian view on everything.

TurboTax has to be up on April 15. Right now, small businesses across the world are depending on QuickBooks Online being available always, right? Understanding the failure points of your software and planning contingencies, whether they be architectural contingencies or runtime contingencies is something that we're constantly doing. We just have to evaluate, what are the top things? Because we can't attack everything. For us, multi-cloud right now has not gone above that threshold.

[0:20:37.2] JM: That peak time, the April 15th peak time, how hard is that to architect for even in the cloud? Is it just a matter of – is it really easy, or are there still little weird edge cases that are tricky to configure?

[0:20:54.9] AB: Yeah. There's always edge cases. I would say that the TurboTax team is amazing. I had the great privilege of being the chief architect of the consumer group for six years, which includes TurboTax. It's an amazing organization. It's an operational machine. In terms of getting ready for tax season and executing during tax season, and the rigor around how we create the content and how we secure the data and everything. It's an amazing software operational machine. I think at some point, we should write a book about it.
When you get into the nuances of hosting on the cloud, it's still complicated, right? Because you have to, despite all the amazing advancements that AWS and GCP and others have made, there's still startup time, right? There's still hardware involved. There's still the bootstrapping time of systems coming up. Tweaking it the right way so that you understand as the growth curve is coming, how early do you scale up? How late do you scale down?

Making sure that you've got all your balancing. You still have networks to deal with. Load balancing continues to be one of the biggest things that we are constantly testing. Do we have the right load balancing rules? Do we have the right? failover rules? Do we have the right health checks in place? What are your third-party dependencies like? You scale yourself up, but then you have these third-party dependencies, can they scale? What are the right patterns that you have in place in terms of containing the blast radius should a software failure happen? That either a degraded performance, or slightly degraded performance, or no customer impact at all, but certainly not a complete outage. There's pretty well understood playbooks on what to do, but it's still a complicated science.

[SPONSOR MESSAGE]

[0:22:37.3] JM: Triplebyte fast-tracks your path to a great new career. Take the Triplebyte quiz and interview and then skip straight to final interview opportunities with over 450 top tech companies, such as Dropbox, Asana and Reddit.

After you're in the Triplebyte system, you stay there saving you tons of time and energy. We ran an experiment earlier this year and Software Engineering Daily listeners who have taken the test are three times more likely to be in their top bracket of quiz scores. Take the quiz yourself any time, even just for fun at triplebyte.com/sedaily. It's free for engineers. As you make it through the process, Triplebyte will even cover the cost of your flights and hotels for final interviews at the hiring companies. That's pretty sweet.

Triplebyte helps engineers identify high-growth opportunities, get a foot in the door and negotiate multiple offers. I recommend checking out triplebyte.com/sedaily, because going through the hiring process is really painful and really time-consuming. Triplebyte saves you a lot of time. I'm a big fan of what they're doing over there and they're also doing a lot of research. You can check out the Triplebyte blog. You can check out some of the episodes we've done with Triplebyte founders.

It's just a fascinating company and I think they're doing something that's really useful to engineers. Check out Triplebyte, that's T-R-I-P-L-E-B-Y-T-E.com/sedaily. Triplebyte, byte as in 8 bytes. Thanks to Triplebyte and check it out.

[INTERVIEW CONTINUED]

[0:24:27.4] JM: The other thing that gives me some comfort in the Amazon space is – so I worked there for eight months. I think my time there overlapped with either a Black Friday, or one of these prime day things. What's nice about Amazon is they have the built-in peak testing, like because they're going to get slammed with traffic on prime day, or on Black Friday, it's cool, well Amazon can at least handle their peak.

[0:24:53.3] AB: Yeah, absolutely. The great test case like you said, for Amazon Web Services is amazon.com. Knowing that nearly all the services have already been either pre-vetted by amazon.com, or actually have been decomposed out of amazon.com become part of Amazon Web Services is definitely a plus. One of the other cool things that Amazon does is there are times where we say, “Well, how does amazon.com use this managed service?” They'll say, “Well, you want to talk to them, then we’ll just physically go talk to them.”

[0:25:24.5] JM: Wow, that's cool.

[0:25:26.0] AB: They'll set those things up.

[0:25:28.1] JM: Do you think of Amazon, or AWS as how you would think about maybe like Spring Framework back in the day?

[0:25:38.2] AB: Yeah. The layer of abstraction from software that Amazon or the cloud in general – I mean, what the cloud is doing is it's slowly commoditizing the stack, right? You start off at the bare bones at compute and storage, compute storage and network, right? At the lowest level, everything is compute storage and network. Then you build on top and you build on top and you build on top.

Yeah. I mean, to us it's about finding that right layer of abstraction and thinking about it is like a Spring Framework? Yeah, absolutely. It is. We want to point our engineers at solving the customer problems that we have as a company. That doesn't mean that we don't solve hard technology problems. We do. We solve really hard technology problems. We talk about some of them already in terms of like a tax engine and so on and so forth. If there are technology problems that we don't need to solve, we don't solve them, right? We let others solve them. I think one the characteristics of a great technology organization is that great engineers want to solve problems that haven't been solved yet.

[0:26:39.9] JM: Agreed. What's interesting about the cloud provider competition, people call competition – this is a gigantic growing space. There's a lot of pie. Pie is growing very quickly. We are moving into this time where you're starting to see services that are differentiated. They are no longer commodity ones, or even you have the question raised as to who is the best at running Kubernetes, or does it actually matter? Does this matter that much? Who has the best service mesh? Have there been any services that have really surprised you that are less of a commodity, where you see the clouds differentiating on their points? Maybe managed Tensorflow, or anything like that?

[0:27:27.0] AB: Got it. I think all the cloud providers are pretty aware of each other. I think that there's reasons why you would pick one cloud provider versus another. For us, Amazon is the right choice. It's a combination of the capability they have, plus the partnership that we've developed with them. At this point, I don't see anything that is differentiating in that way that would get us to say, “Well, we're going to go this way, or that way.”

[0:27:53.4] JM: I'd like to talk about data infrastructure, because Intuit has a ton of data. Both the OLTP and the OLAP queries within Intuit seem really complicated. I would think of an online transaction processing query, like I'm a user, perhaps I'm opening up QuickBooks and I have to load – if you’re QuickBooks, you have to load all these balances for the user. That's an OLTP transaction. It could be complicated, because there's a ton of data.

You could potentially fetch. You could prefetch stuff aggressively. You could be more lazy about it. Also on the OLAP side, the online analytic processing, if you want to do data science on the backend, if you want to present reports to people all the time, you've got a ton of problems there to solve as well. Tell me about how you think about data infrastructure.

[0:28:45.7] AB: Yeah. I think the most important thing to start with in terms of addressing that is those use cases that you called out are all very different. They're very different in terms of the data they require. They're different in terms of their call patterns. They're different in terms of where the data is and how far you have to go to get it.

As part of our data infrastructure, we want to make sure that it's almost like you're trying to support two competing models, right? Because in the simplest sense, “Oh, we're going to have a microservices architecture.” Great. There's books on this, right? Go get the O'Reilly book on microservices architecture, right? Okay, this is how you decompose services and each service has its own database. Then this is how you design its data and da, da, da, da. Then from an OLTP perspective, those systems work fairly well, right? They're performant, they're reliable, you can build failover patterns, you can make them highly available, so on and so forth.

Then you get to the point where you have a data analyst and your analyst wants to do analytics over multiple different data sources, across 15, or 20, or 30 different microservices. Now you're talking about lake architectures and you're talking about analytical store, so you're talking about aggregated data.

I mean, going back to the cloud conversation, in the days before the cloud, you basically had to build expertise, local expertise and managing many different types of data infrastructure, whether it's Oracle databases, or MySQL databases on the OLTP side, to whatever it might be on the analytics side. You first had to build expertise in the data infrastructure on how to set it up, how to operate it, how to monitor it, how to move data through pipelines, so on and so forth.

What we are being very deliberate about is understanding the different call flows that are required for I would say the three primary kinds of workflows; one is the OLTP, the second is the analytics and the third is to support the ever-evolving AI machine learning platform. The technology choices that you make in each of those places has to be different. You have to get really good at moving data around. You have to get very good at understanding taxonomy and ontology. You have to get very good at understanding consistency of things, like data governance, because aside from the CCPA and the European regulations around privacy, privacy is job number one for Intuit; privacy and security based on the data that we have.

It's one thing to secure a database that is accessed by one micro-service, and to secure it and make sure it's private and to implement data governance. To do it now in this distributed environment, you have to enforce consistent data governance across all of those things, no matter what the touch point is, no matter where the data is stored. From a data infrastructure perspective, technology choices are important, how you move data is important. Then finally, how you govern and keep the data private and secure is important. Those are all things that we think about as we design and implement our data infrastructure.

[0:31:58.3] JM: Okay, there's a ton there to unpack. I want to start with the auditability question, because you need to be bulletproof in your ability to go back and audit yourself. If a customer says like, “Hey, this is really weird. Why does this balance happen?” You want to be able to roll back the transaction, the append-only transaction log.

I mean, I've talked to companies that say they're trying to work on machine learning auditability and data auditability and it seems like a very, very hard problem. Many of these companies are dealing with stuff that's way less sensitive than the financial transaction world. You're probably at the forefront of this. You certainly have the incentive to be in the forefront of this. How realistic is it to build auditability into these systems?

[0:32:50.8] AB: I think that's one of the things that for what we talked about before in terms of is the innovation, it's the core part of the Intuit business is to actually be good at that. One of the best examples that I can give you is that five years ago if you go through the process of TurboTax and you either got some deduction, or you didn't get some deduction, or some type of income was taxable and some income was not taxable, or you were itemized versus a standard deduction filer. We couldn't tell you why. The engine would calculate it, but it couldn't tell you why.

One of the amazing innovations that we actually have in our knowledge engine, which backs TurboTax today, is that it can actually explain it to you using your own data. It can say, “You are an itemized filer, because.” Whatever rules that we have in place, whatever data that you gave us that allowed us to make a decision, we can explain it to you.

Explainable knowledge engineering is the step one for us. Then obviously, one of the Holy Grails that's going on right now is explainable ML. There's actually a couple startups out there, many startups out there working on explainable models. Those are areas that we're going into as well, to really make sure that there's full visibility, our customers have full visibility into the decisions that the software makes on their behalf. To understand scenarios of well, what happens? What happens if I remove this? What happens if I change this? Doing those what-if scenarios.

Going back to the original thing that we talk about, about powering prosperity for our customers; if you're affluent, you go to a financial planner, a physical human being who does a lot of what-if scenarios for you. What about everyone else, right? What if the software could do those what-if scenarios for you? What if the software, not only had the auditability and I also have the traceability of what you gave us, what we did and they provide different scenarios? What if we could just show you immediately this is what your refund would be if you took the standard deduction, this is what your refund would be if you did the itemized deduction.

Those are the things that people want – sometimes want that visibility and sometimes they don't, right? That's one of the challenges too is to provide that capability in a way that you deliver it to the customers who want it and you don't deliver to the customers who don't want it. I would say that we are in varying degrees of maturity as it relates to providing that capability, but it is something that is definitely top of mind for us.

[0:35:32.0] JM: Tell me about some different tool selections. We're in this world where we've got a buffet of different options for building this data infrastructure. We can choose cloud databases, snowflake managed proprietary data warehousing, we can use Spark, we can use manage Spark, we can use Kafka, we can use managed Kafka. How do you choose these different data infrastructure decisions?

[0:36:02.0] AB: Yeah. I think this is one of the places where software architecture truly comes in to form. I've always said that the further back you are in the stack, the more important the architecture is. The further front you are in the stack, the less important architecture is. Now when you're in the way, way back and you're inside the data infrastructure, making sure that you build things using certain patterns that allow you to apply the right technology at the right time, I think is more critical than any individual technology choice that you would make.

For example, you brought up Kafka, this is certainly one of the aspects of moving from a desktop engineering culture to an online connected culture, which is synchronous versus asynchronous, right? In your back-end, moving towards asynchronous interactions is absolutely critical. Moving towards messaging infrastructure, topic-based infrastructure for data, absolutely critical. Your ability to create used messaging and streaming to create a curation pipeline that allows you to curate data in the right way to build the right data sources, using the right technology for the right purpose is critical.

We spend quite a bit of time. My team, in fact, directly spends quite a bit of time figuring out what are the right variability points that we want to create in our data infrastructure to support that myriad of technologies that you just described? To make sure that we can actually choose the right technology at the right time. Once you do choose the right technology, certainly one of the considerations is do I have to operate this technology, or can I rely cloud provider to deliver it to me? Certainly, we have a bias towards ones that can be operated for us, as long as we believe they can deliver on the non-functional requirements that we need. Then if we don't, if we can't use that and certainly running our own data infrastructure and in a cloud compute environment is a secondary option.

[0:38:04.0] JM: Right. What are the spaces where you've had to do that? The one that stands out to me is Airflow. There's this – the workflow orchestration tool that seems to have not been really offered as a service much by the cloud providers, to the extent that I understand it. What are the places where you have to spin up your own open source infrastructure on cloud infrastructure?

[0:38:27.1] AB: There's different tech. I don't think I want to call out specific ones, because they change over time. I think that we want to make sure that we're always running on the best tech. For us, the best tech to run is open source that can be managed by a cloud provider. For example, Kafka we could operate our own Kafka, or we can use Kafka as a managed service on AWS, right?

I would say that we're constantly evaluating open source projects as trial standards, standards in experiments, sorry. We refer to them as standards in experiment, to learn and say, "Is it better to do this, or is it better to use the current standard that we have?" In terms of what those things are, certainly any type of data technology that's out there is in any way mainstream, we're trying to figure out whether or not we can apply it.

Anything in the ML world, we're trying to figure out if we can apply it. The reason that I – I probably can't give you specific answers, because we're constantly running those types of experiments. I would say that the bias is towards boy, it'd be nice if I didn't have to worry about this, but I will choose the things that I will worry about and run with the best.

[0:39:44.0] JM: Why does that bias lead you in the direction of open source? For example, you can obviously run Kubernetes as a managed service today. In many cases, that will make a lot of sense for people. There's also these managed container instances, which to me as a developer, these things look better abstractions to work within. Why would I want to manage the Kubernetes orchestration layer when I can hand that off to the cloud provider and just work directly with container instances, like Fargate, or ACI? From the point of view of an architect, how do you evaluate the Kubernetes managed service versus the long-lived container instances, plus AWS lambda architectural models?

[0:40:31.5] AB: Yeah. It's great question. I talked about the model that we have of core versus context and there's actually something in between. The thing that's in between is open source. The times where we would choose something like Kubernetes is where we believe that we have the expertise and need to have the expertise to actually not just consume that open source, but actually contribute to it as well.

If we believe that the pace at which we're moving, we want to have more control over the compute environment, because maybe it's not as mature as you'd like it to be, or maybe it's missing some capability that we need, or maybe it isn't exactly configured the right way, or doesn't provide the configuration in the right way that we want, or we just think that there's a better and faster way to do it, then we'll invest. We view that as an investment.

The things you invest in the most that is core to you, you just build from scratch. The things that are next important to you, you invest in open source. Then the things that you believe you could just consume as commodity, you just acquire through a cloud provider. That's the evaluation that we go through.
We've been on different parts of our journey with AWS, where for example, we started something ourselves, where we actually built it and operated in an AWS. Then over the course of time, AWS matured its capabilities, and so we actually move to consuming it from AWS. We are evaluating that spectrum all the time, but for us it really comes down to control.

[SPONSOR MESSAGE]

[0:42:13.6] JM: When a rider calls a car using a ride sharing service, there are hundreds of back-end services involved in fulfilling that request. Distributed tracing allows the developers at the ride-sharing company to see how requests travel through all the stages of the network. From the front-end layer to the application middleware to the backend core data services, distributed tracing can be used to understand how long a complex request is taking at each of these stages, so the developers can debug their complex application and improve performance issues.

LightStep is a company built around distributed tracing and modern observability. LightStep answers questions and diagnoses anomalies in mobile applications, monoliths and microservices. At lightstep.com/sedaily, you can get started with LightStep tracing and get a free t-shirt. This comfortable well-fitting t-shirt says, “Distributed tracing is fun,” which is a quote that you may find yourself saying once you are improving the latency of your multi-service requests.

LightStep allows you to analyze every transaction that your users engage in. You can measure performance where it matters and you can find the root cause of your problems. LightStep was founded by Ben Sigelman who was a previous guest on Software Engineering Daily. In that show, he talked about his early development of distributed tracing at Google. I recommend going back and giving that episode a listen if you haven't heard it.

If you want to try distributed tracing for free, you can use LightStep and get a free t-shirt. Go to lightstep.com/sedaily. Companies such as Lyft, Twilio and GitHub all use LightStep to observe their systems and improve their product quality.

Thanks to LightStep for being a sponsor of Software Engineering Daily. You can support the show by going to lightstep.com/sedaily.

[INTERVIEW CONTINUED]

[0:44:25.0] JM: As this “service mesh” entity discussion has come to the forefront of software architecture, this idea that there is something new in the idea of having a proxy layer, plus a control plane that does load balancing and canarying and security policy management and so on, how have you responded architecturally to the different offerings, the different visions that people are portraying as the future of service mesh? How is that fitting into your architectural strategy? Or are you just sitting out for a bit?

[0:45:06.5] AB: No. I mean, we're definitely not sitting out at this point. Historically, we've had very much a north-south architecture as it relates to services and gateways and things like that. Then probably about a year and a half ago-ish, we really started to investigate deeply the idea of a service mesh to allow more east-west routing and a little bit more autonomous point-to-point communication, as opposed to going back through the control tier.

We have a team, whose responsibility is to deliver that infrastructure, that we call it our services fabric, for our entire microservices strategy for Intuit. The good news is they were on top of it. They did deep evaluations of what happens if we do it ourselves, what happens if we do some cloud native and what happens if we just consume it from AWS.

We've got certain trials ongoing right now across that spectrum. At this point, we haven't fully made a decision as to what direction we're going to go, but we do believe that service mesh is a very powerful way for us to continue to maximize our spend, to maximize our cloud investment. It's also a way to really maximize scalability, to allow autonomous units of capability delivery to be scaled independently, without a lot of worry about okay, what else do I have to scale when I scale this thing?

I think the team has come up with a pretty good approach for how to meld together the north-south routing and the east-west routing as it relates to services gateways versus services mesh. I'm confident in the next couple of months, we will probably be rolling it out more broadly.

[0:46:55.7] JM: I can see the strain, the tension in how you might approach this architecturally right now, because on one hand, app mesh looks like a great solution for what – the direction you're going in. On the other hand, Istio might win, despite the fact that Istio seems to have been released and promoted a little bit aggressively. It may not quite be ready for production yet, but if the container orchestration wars are any indication, probably Istio is going to win, then maybe does app mesh turn into a hosted Istio, like hard to know, hard to foresee?

People are still running Amazon ECS, I believe. I don't think Amazon ECS is – can easily be migrated to Kubernetes, or maybe it can. I don't really know. Basically, if we map the container orchestration wars to the service mesh wars, well, I mean, it's hard to see Istio losing. I don't know. How do you evaluate – is this the container orchestration wars all over again?

[0:48:02.2] AB: Yeah. I think the interesting thing about disruption, so you can think about things like business disruption of where a company does A well, and then another company comes and they do A well, and so then they compete with each other. Then another company, a third company comes along and they do B and B makes A obsolete, right? That's the disruption that you're like, “Oh, crap. What do I do now, right?”

When you think about Istio and what it's doing and the container war, these are things that are – it would be great if every innovation that came out that you consume would be all just purely additive. Unfortunately, they're not, right? They tend to overlap, because they don't just make the current paradigm better. Sometimes they completely blow up the paradigm and define a completely different paradigm. I think cloud obviously was one of those paradigms.

Intel spent a lot of time telling us that Intel Inside is the most important thing, and then Amazon told us, it doesn't matter what's inside. That's a classic disruption. Is this going to be more like the container? I don't know. I don't know. We want to be aggressive in our pursuit or the best technology to solve the problems that we have, but we also want to be careful to make sure that we don't bet on something too soon.

[0:49:21.0] JM: Are you experimenting with Knative at all?

[0:49:23.6] AB: We are. We actually, about a year and a half ago, we bought a little company called Applatix. Applatix was building some infrastructure tooling around Kubernetes. We've developed a team that has quite a bit of expertise around Kubernetes. Now as we think about Knative and what it can do for us, as I said before, we have a strong bias towards open source. We have a strong bias towards technologies that we know, especially the ones that we contribute in open source. Knative is one of the places where we're currently running experiments. Yes.

[0:50:00.6] JM: How does it look as a technology?

[0:50:03.5] AB: I think it's at the stage right now where when you talk about the wars, like technology wars, so there are people on the Knative side and there are people not. I think it's early, but I think that we are at a place where we at an influencer stage on it. We're pretty confident that it can get to a place where it can become part of our portfolio.

[0:50:27.7] JM: You think it's legit? You think this moving an auto-scaling layer to the open source world, the Knative belief set that there is no function as a service, there is no container as a service, these two things are the same along a gradient, you believe this may be the reality?

[0:50:48.4] AB: It could be. It could be. Certainly, Amazon has a very strong perspective on serverless and containers. I'll tell you that this is certainly one of these places where I think it's too early in the game to make any type of specific choice of what in fact is what we want to lean into. When we think about standards at Intuit, we talk about it as fixed, flexible and free. Fixed is there's exactly one standard and thou shalt use it. Flexible is just a couple things to choose from. Free is hey, engineers go out there and should go change the world.

I think this is something that certainly, we don't know enough about to declare one way or another, in terms of do we think that it exists as a spectrum, or is it a little bit more of a fixed paradigm? We're actively running experiments to figure that out.

[0:51:40.6] JM: It's really cool that you do that, because I think you could just as easily say, "Yeah. Look, we're just going to sit out and wait for the managed version of all of this." How do you assign resources, to what is essentially like, R&D, or platform engineering experiments, or under what part of the balance sheet you would put that, the engineering balance sheet?

[0:52:06.8] AB: Yeah. One of the things that Intuit does very explicitly is manage, is investment as a portfolio. We understand how much are we spending on growing the business, how much are we spending on building the future of the business and how much are we spending on technology hygiene. Included in hygiene would be futures and experimentation.

We have entire teams whose responsibility is just to think about tech futures. In fact, one of them is called tech futures. Then there are also sub-budgets for many of the infrastructure teams to be constantly evaluating what's new and to be figuring out how should we be evolving our stack. We're really passionate about trying to deliver the best possible development environment for our engineers.

In order to do that, we want to make sure that we're on top of this. This started with Tayloe Stansbury, who was the CTO of Intuit for the past 10 years and now certainly continuing on with Marianna Tessel, who's the CTO starting in January, where we make sure that we invest to be a very healthy technology company. The investment to be a very healthy technology company means, sometimes you're spending money on tech simply to be better at tech, which will then pay huge dividends and delivering the technology to deliver experiences and services to our customers.

It's a really hard thing to do. It's a hard thing to justify. This isn't a direct equation, X equals a plus B, right? There isn't a clear causal relationship that you can show. This is just the experience of engineering leadership, the ability as technology leaders to communicate with the business side of the house and to say, “Hey, listen. We have to invest some of this to continue to be a healthy technology company.” In general, Intuit knows this. Intuit as a company, it's one of – like I said, one of the reasons I've been around for 20-plus years, or not 20 plus, almost 20 years, is that it's – Intuit as a company is constantly reinventing itself.

There's no reason why Intuit the desktop software company should have become Intuit the SaaS company, except for the fact that we disrupted ourselves. Actually, the business gave itself permission to cannibalize the existing customer base with the new product. When it comes to technology, we have to do a similar thing. That portfolio management goes on.

[0:54:39.8] JM: Do you also do a 20% time thing as part of the bottoms up speculative investments, or do you just say, “Look, the core product is so – we have so much to do in the core product. Let's just like, when it comes to new features and new stuff, let's keep it in a product – let's sequester that in the product development side of things, also because your data is sensitive, maybe you don't want people really hacking on 20% style projects in a data sensitive environment.” How does that fit into your speculativeness?

[0:55:15.2] AB: Yeah. I mean, part of our – just to make sure where you're going on this. Part of our budgeting process is specific time for engineers to work on hygiene, to make systems better. Then we also have engineering events, a week-long engineering events, where basically we tell the engineers, go do something cool. There's almost no constraints. The constraints are things like, “Well, you have to have consented access to customer data.” You can't go access customer data. You can build an experience, or service, release it to production that requires a customer to consent access to their data. We're very good at generating mock data that looks like customer data, but is not, if you want to build these types of things.

We're firm believers that our engineers need to continue to hone their craft. Software engineering is a craft, much more than almost any other type of engineering activity, right? If you're an engineer designing bridges, you generally don't go in your backyard and build test bridges too much, right? Yet as a software engineer, in order to stay current and up-to-date and on top of things, you need to constantly be honing your craft.

We try to make sure that there are activities and time and place that the engineers can hone their craft, to become better engineers, to learn new software languages, to learn new technologies. These innovation events, these engineering, for lack of a better word, hackathons are great ways to do that.

[0:56:50.9] JM: Awesome. This has been a really interesting conversation. I want to end on, I guess a far-flung question. What is an Intuit product that you wouldn't be able to build today, but might be possible in 10 years?

[0:57:03.9] AB: Well, that's a really good question. I think what it really comes down to is the data that we talk about, that Intuit as a company is as we are becoming more and more of a platform, a data company where we use the experiences that we have to collect data on behalf of the customer. To be clear, we have very, very strict data stewardship principles. Number one in the data stewardship principle is it's not our data, it's the customer’s data.

Once we do have that data, our ability to derive insights for those customers is ever-growing. For example, our ability to predict cash flow for small businesses is ever-growing. One specific example of something that we released actually last year, which will give you an idea of the kinds of things that we can do in the future is that there are a number of small businesses.

One of the biggest problems that small businesses, early small businesses have is they don't have the capital to grow. What do they do? They go to a bank. They don't have any track record, so the bank unfortunately can't lend to them. With the data that we collect through QuickBooks, through the financial transactions from their bank and through the transactions that they put into QuickBooks, we've developed models that allow us to assess risk that are rate better than what the financial institutions can do.

We have started creating micro-loans direct lending to small businesses. So much so, that the banks now are saying, “That's great. Can we get in on that too?” That is just the smallest sliver of the type of insights that we can build on top of the data that our customers have allowed us to steward on their behalf.

In the future, you can think about things like, financial, or cash flow projections, so we can allow our customers to make decisions that are weeks ahead of time. They can make decisions before they run out of money to actually not run out of money. That can be true about a small business who has a big order coming in and understanding how much inventory do they need to acquire and therefore, what implication does that have in their cash flow. It can include families who are trying to understand what order do I pay my bills? How do I make more money, as the world becomes more and more of a gig culture? How many hours should I work at Uber to actually be able to pay my bills this week? Those are the types of projections and insights that we can provide in the future, that we can't quite yet do today, but it's definitely right around the corner.

[0:59:38.8] JM: Alex, thanks for coming on the show. It's been great talking.

[0:59:41.6] AB: Yeah, it's been great talking to you too.

[END OF INTERVIEW]

[0:59:47.3] JM: GoCD is a continuous delivery tool created by ThoughtWorks. It’s open source, it’s free to use and GoCD has all the features that you need for continuous delivery. You can model your deployment pipelines without installing any plugins. You can 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 cloud native project.

With GoCD on Kubernetes, you define your build workflow, you let GoCD provision and scale your infrastructure on the fly and GoCD agents use Kubernetes to scale as needed. Check out gocd.org/sedaily and learn how you can get started. GoCD was built with the learnings of the ThoughtWorks engineering team, and they have talked in such detail about building the product in previous episodes of Software Engineering Daily. ThoughtWorks was very early to the continuous delivery trend and they know about continuous delivery as much as almost anybody in the industry.

It’s great to always see continued progress on GoCD with new features, like Kubernetes integrations, so you know that you’re investing in a continuous delivery tool that is built for the long-term. You can check it out for yourself at gocd.org/sedaily.

[END]